UNIVERSITAT POLITÈCNICA DE CATALUNYA

Escola Politècnica Superior d'Enginyeria de Manresa

Disseny i construcció d'una maqueta de control basada en un dron

11 d'octubre de 2019

treball de fi de grau que presenta

Pau de las Heras Molins

en compliment dels requisits per assolir el

Grau d'Enginyeria en Sistemes TIC

Direcció: Dr. Eng. Jordi Bonet

Al tutor del meu projecte, el Jordi Bonet.
Per ensenyar-me que la qualitat més important d'un enginyer és l'actitud,
i que el control és molt més complicat del que sembla.

Resum

La creixent presència dels vehicles aeris no tripulats o drons en el mercat de consum actual demostra l'increment en popularitat que ha experimentat aquesta tecnologia en els últims anys, cada vegada més estesa i explorada pel que fa a les possibilitats que ofereix. Mentre que la recerca actual en aquest sector sovint se centra en les múltiples aplicacions que poden tenir els drons, aquest treball de fi de grau en canvi pretén fer un estudi del seu funcionament intrínsec, concretament fixant-se en els controladors de vol, els "cervells" dels drons que implementen les estructures que fan possible que volin en primera instància. Els coneixements obtinguts en aquest àmbit serviran per dissenyar i construir una maqueta de control basada en el funcionament d'un dron i per acabar implementant un sistema de control que permeti dirigir el seu vol. Al final, l'objectiu del projecte és que aquesta maqueta sigui una plataforma d'aprenentatge que permeti portar a la pràctica conceptes sobre teoria de control. El treball, per tant, gira al voltant de tres eixos: el disseny i la construcció física de la maqueta del dron simplificat per una banda, que serà la planta del nostre model, el desenvolupament de les eines de monitoratge del sistema i el disseny i l'anàlisi de les estructures de control que ens permetran governar aquesta planta.

Abstract

The growing presence of unmanned aerial vehicles or drones in current consumer market proves the increase in popularity that this technology has experienced in recent years, becoming more widespread and explored when it comes to the possibilities it can offer. While current research around drones usually focuses in the multiple applications drones can have, this bachelor thesis aims instead at studying their intrinsic functioning, especially regarding their flight controllers, the "brains" of drones that implement the structures that make it possible for them to fly in the first place. The knowledge obtained in this topic is used to design and build a model based on the operation of a drone and to implement a control system that is able to command its flight. In the end, the goal of this project is to have this model serve as a learning platform that allows to put into practice the principles of control theory. Therefore, the thesis revolves around three main axes: the design and construction of the prototype of a simplified drone, that will act as the plant of our model, the development of the monitoring tools for the system and the design and analysis of the control structures that will allow us to command this plant.

Índex

1 Introducció

1.1 Motivació

Els aparells coneguts pel nom de drons són un tipus de vehicles aeris no tripulats (en anglès, *Unmanned Aerial Vehicle* o UAV) que s'han popularitzat enormement especialment durant l'última dècada (tal com es pot observar a la figura 1.1) constituint-se com un sector creixent del mercat a mesura que els avenços tecnològics n'han anat ampliant les aplicacions.

Actualment aquest sector continua creixent, i l'augment de presència tant dels drons comercials (els utilitzats com a eines com a part d'un negoci) com dels drons recreatius (els consumits com a producte recreatiu, com a *hobby*) ha propiciat encara més la recerca de noves aplicacions per a aquests dispositius, fet que al seu torn ha continuat impulsant-ne el creixement.

Dins aquest cicle de creixent interès que es remunta a principis del *s. XX* amb el naixement del concepte de drons dins l'àmbit militar, tal com s'explora a [**sciencedirect_2017**], molts cops una pregunta que no rep tota l'atenció que caldria és "com funcionen els drons exactament?". Aquesta pregunta, que no és difícil intuir que té una resposta molt més complexa i àmplia del que sembla, pot dur-nos a explorar una de les bases més importants per al funcionament dels drons: els sistemes de control que permet que aquests volin en primer lloc.

Sovint els detalls sobre el funcionament d'aquests sistemes de control, i en general del *software* que incorporen els controladors de vol (els ordinadors centrals que governen l'operació dels drons), poden ser difícils de trobar o senzillament parteixen d'uns coneixements relativament avançats sobre el tema que fa difícil que es pugui utilitzar el reclam que suposen els drons com a via d'entrada per aprofundir en els conceptes teòrics subjacents, desaprofitant una oportunitat molt atractiva des d'un punt de vista acadèmic.

És arran d'aquesta situació que neix la motivació per aquest treball de fi de grau. Amb la intenció d'explorar la dimensió amb més interès acadèmic i docent que ofereixen els drons, una tecnologia atractiva que encara està en evolució i que engloba les diferents branques de les TIC, aquest projecte busca inspirar-se en aquests aparells per crear una plataforma que permeti als professors ensenyar, i sobretot als estudiants aprendre, els conceptes teòrics de fons que han permès l'evolució dels drons d'una manera més pràctica i engrescadora.

1.2 Objectius i abast

A l'hora de plantejar els objectius que persegueix aquest treball de fi de grau inicialment es van plantejar definir tres eixos al voltant dels quals giraria aquest: per una banda tenim el disseny i la construcció física d'una maqueta de control basada en un dron que actuï com a planta del sistema, el desenvolupament de les eines que permetin monitorar i configurar aspectes d'aquest des d'una "estació terrestre"[1] (que s'executarà des d'un ordinador), i finalment l'estudi

[1] Al llarg de la memòria es fa referència a aquesta estació de control terrestre, o senzillament estació, com a aplicació de comunicació, monitoratge i control del dron que s'executarà des d'un ordinador.

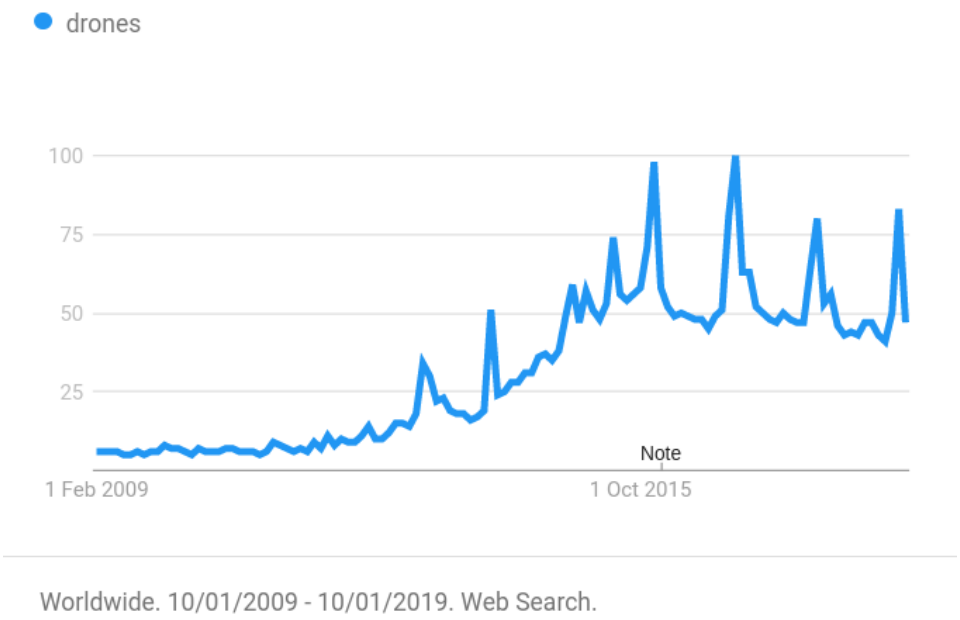

Figura 1.1: Creixement del nombre de cerques a *Google* del terme "drones" durant l'última dècada. Font: *Google Trends*.

d'algunes de les estructures de control més típiques per tal d'implementar-les i poder governar el vol del dron.

També és important considerar que, tal com es comenta al capítol anterior, la motivació de la qual neix la idea d'aquest projecte és la de crear una eina docent que permeti observar, experimentar i en definitiva aprendre tot posant en pràctica alguns dels conceptes relacionats amb la teoria del control que s'apliquen, entre d'altres, als drons.

Amb els tres eixos que s'han comentat, i la visió que es deriva de la motivació per aquest projecte, doncs, s'han definit els objectius del treball de fi de grau tal com segueix:

1) Construir una maqueta de control basada en un dron

2) Desenvolupar el conjunt d'eines *software* que permetin monitorar i controlar el comportament del sistema des d'una estació terrestre

3) Implementar i estudiar, mitjançant les eines desenvolupades, un sistema de control funcional per a la maqueta construida

Aquest projecte pretén abastar, en definitiva, tot el procés de creació de la maqueta de control i la utilització d'aquesta eina per a estudiar i aprendre sobre el funcionament d'un sistema de control que ens permeti governar el dron. D'aquesta manera es pretén il·lustrar les possibilitats que ofereix la plataforma creada per obrir la porta a què s'utilitzi amb aquesta finalitat, la d'estudiar i aprendre sobre teoria de control.

1.3 Organització de la memòria

La memòria del treball consta de tres blocs principals, el primer dels quals és el disseny i la construcció de la planta, on s'expliquen els detalls relacionats amb els aspectes mecànics i amb l'electrònica de la maqueta física. Seguidament tenim el desenvolupament del programari, capítol en què es descriu l'estructura de mòduls i en general el funcionament del *software* implementat com a part del projecte. Finalment, l'últim dels tres grans blocs de la memòria és el que parla sobre el control del sistema creat, començant per una breu introducció sobre conceptes relacionats amb la teoria de control, passant per la simulació dels models de la planta i el controlador i acabant amb l'avaluació del funcionament del control a la planta real.

La memòria acaba amb el capítol dedicat a concloure el projecte i a presentar les possibles línies de treball futur que se'n deriven.

2 Disseny i construcció de la planta

Al llarg d'aquest capítol es descriuen tots els aspectes que tenen a veure amb el disseny i la construcció de la planta física del nostre sistema, que en algunes ocasions també anomenarem maqueta o, senzillament, quan ens referim a l'element mòbil, dron.

El motiu pel qual tota l'estona s'ha dit que la maqueta estarà només basada o inspirada en un dron és per un seguit de motius o decisions en el disseny que han marcat que la maqueta només mantingui la semblança en alguns aspectes amb un dron "real", entenent com a tal el més típic exemple de dron comercial o recreatiu; si bé es poden trobar tota classe de drons pel que fa al nombre de rotors, la forma del xassís, la mida o el pes, entre d'altres, es pot dir que el dron per excel·lència al mercat és el quadrotor (en anglès, *quadcopter*), que es caracteritza principalment pels quatre rotors col·locats en forma de creu que el propulsen per l'aire, i pel controlador de vol que incorpora al centre, tal com es pot observar a la figura 2.1.

La versió final del nostre dron presenta múltiples diferències respecte d'un quadrotor de referència com el descrit. Algunes d'aquestes diferències, que en la majoria de casos suposen simplificacions en el disseny respecte de les característiques dels drons reals, són el resultat de diverses consideracions que s'han tingut en compte per tal d'acabar tenint una plataforma que tot i ser òbviament menys funcional que un dron, sigui útil per a la docència donada la seva major simplicitat tot sent representativa del funcionament de les estructures de control que incorporen aquest tipus de vehicles.

A continuació, doncs, es detallen totes les decisions de disseny que s'han pres i el procés de creació de la maqueta que ha dut fins a la versió final. L'explicació s'ha dividit d'acord amb els diferents elements que conformen la maqueta: els elements de tipus mecànic o físic per una banda, i per l'altra els diferents components que formen part l'electrònica del dron.

2.1 Aspectes mecànics

2.1.1 Suport

Una de les primeres diferències que es poden destacar entre la nostra maqueta i un dron real és el fet que mentre que un dron es diu que té sis graus de llibertat (en anglès, *Degrees of Freedom*, o DOF), sent aquests els paràmetres que permeten caracteritzar la seva posició i orientació en qualsevol moment i punt de l'espai, en el cas de la maqueta resulta que només tenim un grau de llibertat: el dron simplificat només es pot desplaçar al llarg de l'eix vertical, sense poder girar sobre si mateix, mentre que típicament un dron es pot moure i rotar en qualsevol direcció.

Si bé es podria pensar que aquesta simplificació pot resultar ser massa dràstica, passant de sis a un grau de llibertat, la realitat és que controlar el moviment d'un dron amb un sol grau de llibertat ja pot suposar per si mateix un repte, i experimentar sobre els efectes d'aquest control en una planta com aquesta, tot i simplificada respecte d'allò que imita, resulta molt interessant des del punt de vista acadèmic, que no deixa de ser l'objectiu principal del projecte.

Així doncs, coneixent el tipus de moviment que pot realitzar el dron de la nostra maqueta,

Figura 2.1: El *Phantom 3 4K*, un popular model de quadrotor de la marca xinesa DJI. Font: *Unsplash*.

queda clar que un element clau d'aquesta ha de ser el suport o guia que limiti el desplaçament del dron sense dificultar-ne el moviment al llarg de l'eix corresponent a l'únic grau de llibertat que té.

La primera elecció per a aquesta guia o suport que hauria de tenir la maqueta va consistir en un sistema de carro i guia lineal, un tipus de component mecànic que es caracteritza per tenir la capacitat de desplaçar càrregues amb precisió al llarg d'un eix o guia que es pot collar a un altre suport. Concretament, el parell carro/guia que es va escollir va ser el de la figura 2.2, més específicament el models amb referència `HGW15CC` (ambdós elements apareixen amb més detall al catàleg de *Hiwin* que es pot trobar a [**hiwin**]).

Tot i que inicialment aquesta semblava que podria ser una bona opció, després d'adquirir la guia i el carro es va observar que l'alta precisió del sistema, que de fet és una de les característiques més importants d'aquest tipus de sistemes, s'acabava traduint en un elevat fregament que, juntament amb el pes considerable del carro, van dur a desestimar aquesta primera aproximació a causa de les dificultats per a desplaçar el dron en ambdós sentits.

En una segona iteració del procés d'elecció dels materials a utilitzar, es va optar per utilitzar com a substitut de la guia anterior un perfil d'alumini com el que es pot observar a la figura 2.3. Després de fer proves amb diversos materials per avaluar les forces de fregament d'aquests en contacte amb l'alumini del perfil, es va determinar que una opció interessant seria la d'utilitzar peces de plàstic impreses en 3D, ja que el fregament entre aquests dos materials era molt menor al que s'havia tingut utilitzant la guia *Hiwin* al mateix temps que oferia més flexibilitat pel que feia al disseny i fabricació de les peces, fets que vas acabar fent decantar la balança a favor d'aquesta alternativa.

Per tal de mantenir en posició vertical aquest perfil d'alumini la maqueta incorpora un peu

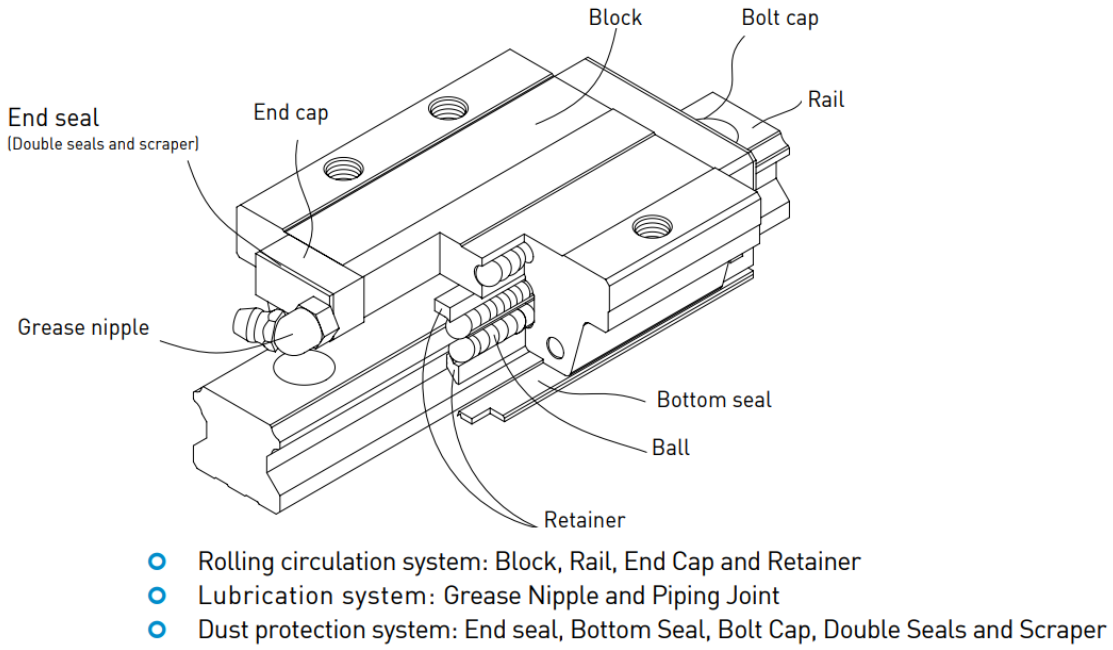

Figura 2.2: Diagrama del muntatge d'una guia lineal amb carro del tipus HG (amb rodaments de precisió) de la marca *Hiwin*. Font: Catàleg *Hiwin*.

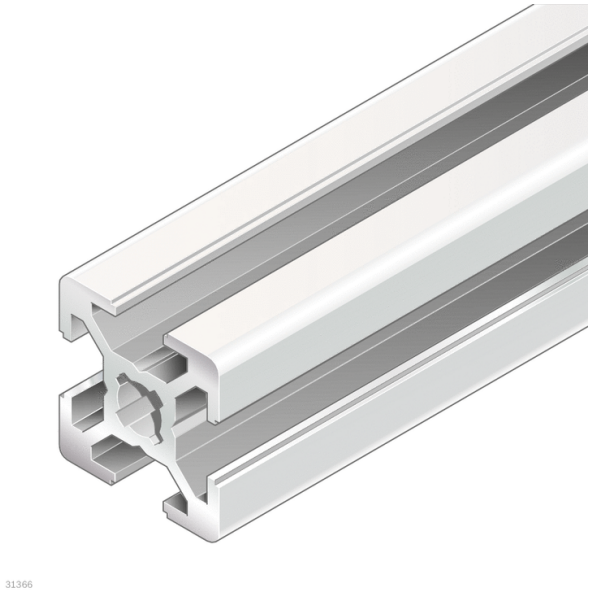

Figura 2.3: Diagrama d'un perfil d'alumini estàndard com el que s'ha utilitzat per al suport del dron. Font: *Rexroth*.

que subjecta el perfil de manera que no s'inclini, i que al mateix temps té un pes prou elevat per a evitar vibracions excessives de la barra.

Finalment, per tal d'esmorteir les caigudes i, en general, els xocs que pogués patir el dron la maqueta també incorpora uns retalls d'espuma subjectats als extrems del perfil per mitjà d'unes pinces (a la part inferior, on el retall és més gran, ja que els xocs potencialment tindran més velocitat aquí) i d'un cargol collat al perfil amb una femella especialitzada.

Un cop definit en què consisteix el suport vertical de la maqueta, el següent pas consisteix a explorar el disseny de l'esquelet o xassís del dron en si, que òbviament es va desenvolupar tenint en compte les restriccions imposades pel disseny final del suport.

2.1.2 Esquelet

Com s'ha comentat, després de valorar l'opció del carro i la guia lineal com a suport vertical de la maqueta, aquesta es va acabar descartant per culpa dels problemes relacionats amb l'elevat fregament i el pes. Així doncs, buscant una alternativa que oferís una solució a aquests problemes, es va arribar a la conclusió que una opció vàlida era la d'imprimir l'esquelet del dron en 3D de manera que s'adaptés al perfil d'alumini escollit com a opció final per al suport.

La tecnologia d'impressió permet fabricar estructures amb una solidesa i robustesa considerable (si es dissenyen les peces adequadament), però d'una manera molt més econòmica i flexible i obtenint uns resultats prou bons fins i tot sense requerir uns coneixements avançats, permetent dissenyar les peces a mida mitjançant programes d'ajuda al disseny amb ordinador (en el nostre cas, s'ha utilitzat *OpenSCAD*). A més, el resultat és un esquelet molt més lleuger, ja que no s'utilitza el carro de la marca *Hiwin*, molt més pesat que una peça impresa en 3D amb una funció semblant.

Des d'un primer moment es va tenir en compte que un aspecte a considerar a l'hora de dissenyar l'esquelet del dron, que podria resultar útil de cara a facilitar i agilitzar-ne el prototipatge, seria la modularitat del disseny. Amb aquesta idea en ment, es van anar dissenyant els diferents components de l'esquelet de manera que fos possible de substituir-ne qualsevol d'ells en un moment determinat (ja fos per un canvi en el disseny, una nova versió o per renovar la peça a causa del desgast) sense necessitat de tornar a fabricar la resta, guanyant temps i estalviant en recursos.

Pel que fa a l'esquelet o xassís del dron, podem distingir tres elements principals que s'expliquen a continuació.

Camisa o element lliscant del conjunt

L'element que ha d'estar en contacte permanent amb el perfil d'alumini, i que per tant experimenta el fregament que s'oposa al moviment del dron, és la que anomenem "camisa" de l'esquelet: es tracta d'una peça que envolta el perfil d'alumini i s'hi ajusta amb un marge suficient per a no tenir joc (si en tingués, la peça es podria quedar encallada per culpa d'una certa inclinació) tot tenint el mínim fregament possible, que permeti que el conjunt arribi a caure pel seu propi pes.

La manera com es va arribar a determinar el valor òptim per a aquest marge va ser a partir de diverses iteracions, fabricant peces amb marge variable fins a trobar aquell que millor complís amb els paràmetres descrits. Cal destacar que al llarg de les proves fetes durant la realització del projecte es va detectar que el constant fregament entre el plàstic i l'alumini dóna lloc a un desgast progressiu de la peça impresa en 3D en aquelles zones on el contacte és major. Per

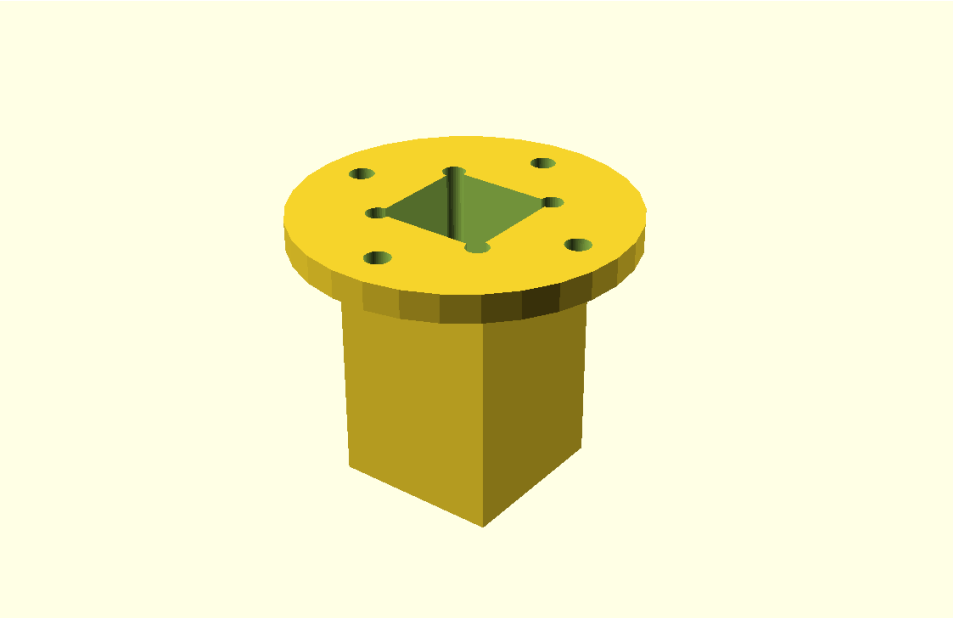

Figura 2.4: Model 3D de la camisa de l'esquelet del dron.

aquest motiu, es van imprimir diverses còpies de la versió definitiva de la peça, que es pot observar a la figura 2.4, per a tenir recanvis periòdics.

La camisa també incorpora una plataforma amb forats que permeten unir aquesta peça amb la resta de components de l'esquelet del dron per mitjà de cargols i femelles. Aquest mateix esquema, tal com ja s'ha comentat, s'ha utilitzat en les altres peces per permetre la modularitat que es perseguia amb aquest disseny.

Braços

Aquesta peça és la que ha de subjectar els motors que propulsin el dron fent rotar les hèlices per generar la corresponent força d'aixecament. Per aquest motiu aquest component de l'esquelet és el que està subjecte a més forces de flexió (ja sigui degut a la força que fan els motors quan aquest giren o a causa de la inèrcia que aquests duen en el moment de l'aterratge), i per tant a l'hora de dissenyar-lo va caldre posar especial atenció al gruix de la peça. També es va tenir en compte en el moment del disseny les mesures dels motors utilitzats per al dron (que es detallaran més endavant) per a poder-los subjectar de manera eficaç, sense exercir pressió innecessària en cap part d'aquests.

Igual com passava amb la camisa, va ser com a resultat de diverses versions fallides que es va arribar a tenir el disseny final, que es pot observar a la figura 2.5, amb el gruix suficient per a no trencar-se per culpa de l'acció de les forces però que comporti menys material i, per tant, també menys pes.

Suport per al controlador de vol

L'última peça que conforma l'esquelet del nostre dron és el suport per al controlador de vol. Aquest component essencial de tot dron normalment s'ubica al centre de gravetat de l'esquelet, ja que d'aquesta manera els sensors que incorpora també poden aportar informació més

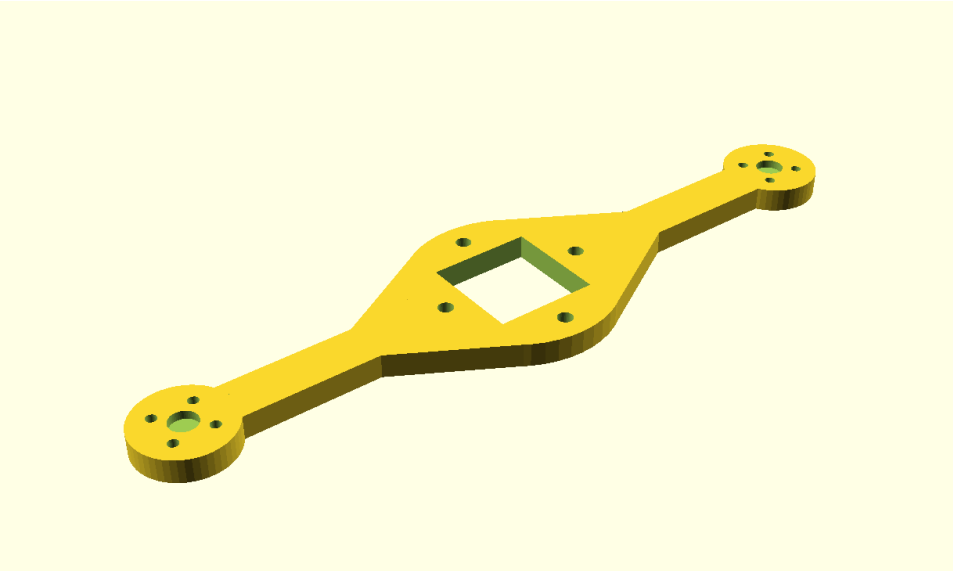

Figura 2.5: Model 3D dels braços per als motors.

rellevant i fiable: per exemple, en el cas d'una IMU o *Inertial Measurement Unit* (unitat que pot incorporar sensors d'acceleració i velocitat de rotació, entre d'altres) la informació més interessant pel que fa a la inclinació i la rotació d'un cos és la que s'obté des del centre d'aquest (d'aquesta manera es redueix l'impacte en la mesura que poden tenir les vibracions o acceleracions a les parts més allunyades del cos principal).

En el cas de la nostra maqueta, però, aquesta configuració típicament utilitzada no va resultar possible, ja que tal com s'ha explicat anteriorment l'esquelet del dron envolta completament l'eix al llarg del qual es desplaça. Així doncs, finalment es va optar per situar el controlador de vol (que en el nostre cas, com es detalla més endavant, es tracta d'una plataforma *Arduino*) a un lateral de l'esquelet però així i tot centrat per tal de mantenir l'equilibri de tot el muntatge, tal com es pot observar a la figura 2.6. La manera com la placa *Arduino* se subjecta al suport és per mitjà d'unes peces metàl·liques especialment dissenyades per a fer de "potes" de la placa, que al seu torn estan collades a la peça impresa en 3D que incorpora els forats als llocs adequats.

Igual com passava amb els braços pels motors, en aquesta peça es va haver de tenir en compte el gruix ideal per a evitar que es trenqués a conseqüència de xocs o acceleracions massa elevades. Amb aquest objectiu, i després d'algunes versions fallides, es va acabar augmentant el gruix i incorporant a la peça una quantitat extra de material a l'angle recte que forma aquesta per a reforçar la paret.

Amb aquesta peça queda completat el disseny final per al dron de la maqueta, que un cop muntat té l'aparença la que es pot observar en el renderitzat en 3D de la figura 2.7.

2.1.3 Motors

Els motors d'un dron són l'element que transforma l'energia elèctrica que alimenta l'aparell en energia mecànica, que a través de les hèlices que s'hi connecten s'acaba traduint en la força d'empenyiment, anomenada *thrust force* en anglès, que fa volar el dron. Com és lògic,

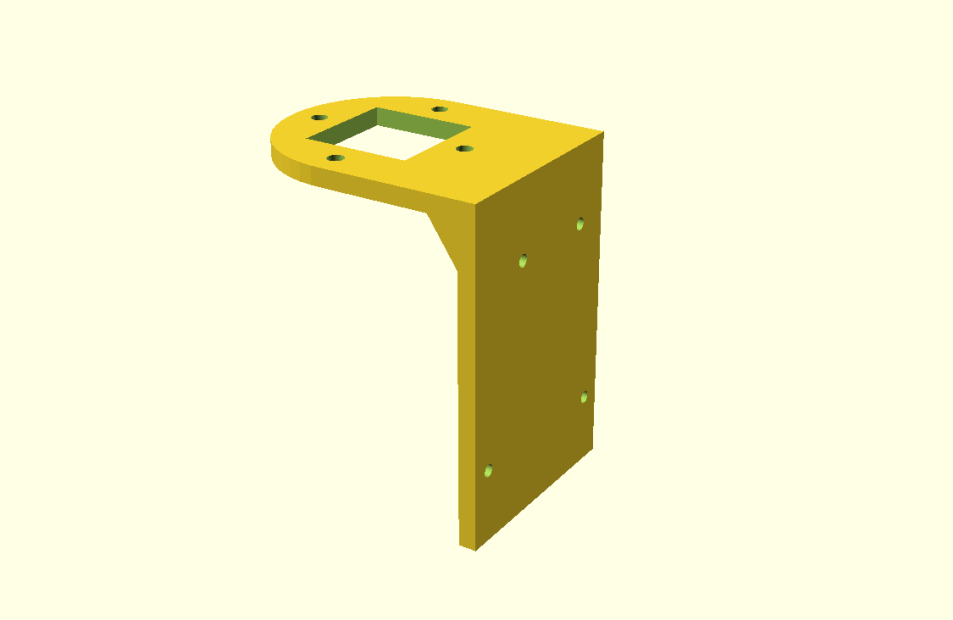

Figura 2.6: Model 3D del suport per al controlador de vol del dron, en el nostre cas una placa *Arduino* motors.

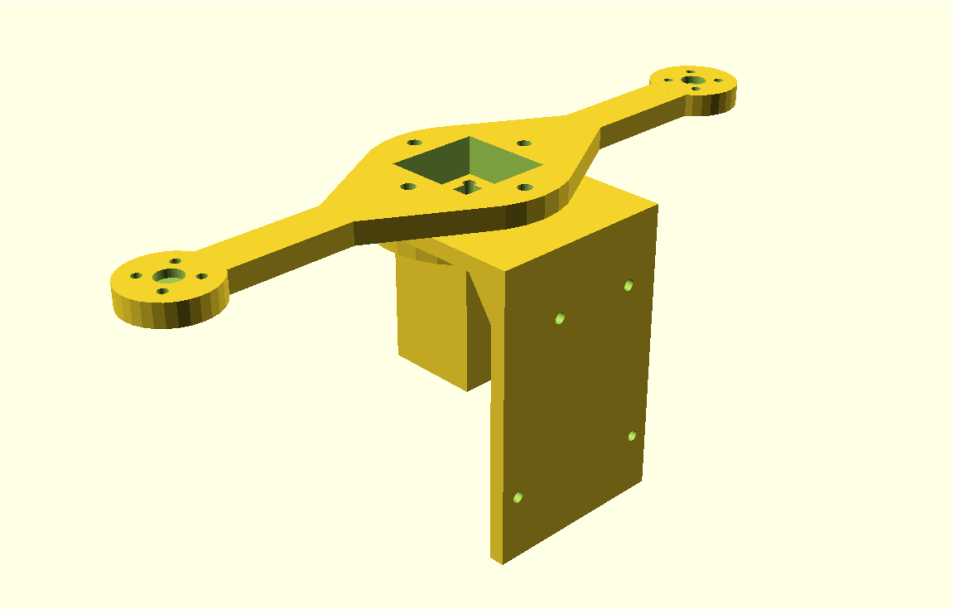

Figura 2.7: Renderitzat 3D del disseny final del dron.

Especificacions	Ion 2207 2500KV	Turnigy C2206 2300KV
Configuració imants	12N14P	14N12P
KV	2500 KV	2300 KV
Cel·les Lipo	3-4 S	2-4 S
Corrent Màxim	34,50 A	23,00 A
Potència Màxima	438,00 W	310,00 W
Pes	29,80 g	31,00 g
Diàmetre	28,40 mm	28,00 mm
Alçada (sense eix)	17,00 mm	19,50 mm

Taula 2.1: Especificacions dels dos models de motor de drons estudiats.

d'uns elements com els motors d'un dron s'espera una eficiència màxima que permeti obtenir el màxim de força tot minimitzant el pes del motor en si, el consum energètic d'aquest i les pèrdues en forma de calor que pugui tenir. És per aquest motiu que diversos tipus de motors poden tenir aplicacions diferents donades les característiques que es busquin —en el cas dels drons, normalment s'utilitzen motors anomenats *brushless*, sense escombretes, que s'alimenten a partir de corrent continu que el *driver* del motor converteix en corrent altern que fa girant el motor manipulant els camps magnètics que s'hi generen.

Aquests factors d'eficiència i d'altres són els paràmetres que caracteritzen els motors de drons, i els que permeten dimensionar els motors a utilitzar en un projecte en concret. En el nostre cas, d'acord amb una aproximació del pes que tindria el conjunt d'elements que conformen el dron, es van escollir dos models de motors diferents disponibles al mercat i se'n va estudiar el funcionament per a decidir quin seria el que s'utilitzaria finalment.

Les característiques més rellevants per al nostre cas dels dos models de motors estudiats, extrets de les especificacions, són les que apareixen a la taula 2.1.

Un paràmetre interessant a l'hora de dimensionar aquests motors és els "KV" del motor, una notació molt estesa entre venedors d'aquest tipus de motors que equival a les revolucions per minut (RPM) sense càrrega que pot assolir el motor per cada volt de tensió aplicat a l'entrada. La tensió màxima que se li pot aplicar, i que per tant permet conèixer les RPM màximes del motor, es pot obtenir multiplicant el valor nominal de tensió d'una cèl·lula de tipus *Lipo* (basada en polímers de liti), equivalent a 3,70 V, pel nombre de cèl·lules recomanat (per exemple, *3-4S* indica que es recomana utilitzar bateries formades per 3 o 4 cèl·les *Lipo*). D'aquesta manera, tenim que en el cas del motor de la marca *Ion* les revolucions per minut màximes són de

$$2500 * 4 * 3.7 = 37000\ rpm$$

i en el cas del motor *Turnigy* són de

$$2300 * 4 * 3.7 = 34040\ rpm$$

La força d'empenyiment que poden arribar a realitzar els motors depèn de la velocitat de rotació de les hèlices —a més velocitat de rotació, més força d'empenyiment. Tot i que òbviament aquesta relació es podria definir amb molta més precisió considerant l'elevada complexitat dels

conceptes físics relacionats, en el nostre cas és suficient conèixer aquesta relació per a determinar que l'opció més adequada possiblement és aquella que proporcioni més potència, i per tant al final més velocitat de rotació.

Tal com s'explica més endavant, en un moment determinat es va decidir que la manera com s'alimentaria el dron seria per mitjà d'uns cables connectats a una font d'alimentació fixa connectada a la xarxa. Això implica que l'eficiència energètica de la maqueta no és un factor tant limitant en el nostre cas com ho és en els drons reals, que busquen aprofitar al màxim la càrrega de la bateria que incorporen. Per aquest motiu, després de confirmar que efectivament les velocitats de rotació que pot assolir el motor *Ion* són superiors a les que pot assolir el motor *Turnigy* per mitjà de mesures de l'espectre de freqüències auditives (com s'explicarà més endavant), es va optar per la primera opció, ja que pot proporcionar major potència —i per tant més facilitat per a moure el dron, una qualitat que a priori sembla raonable considerar útil.

Un altre aspecte interessant a comentar pel que fa als motors del nostre dron és que s'ha de tenir en compte a l'hora d'instal·lar-los l'esquelet que és necessari que el moment angular resultant de sumar els que tenen els dos motors quan aquests giren ha de ser zero. Això s'aconsegueix tenint els dos motors girant en sentits oposats, un en sentit horari i l'altre en sentit antihorari —si no es fes així, el moment angular resultant faria que tot l'esquelet tingués tendència a girar donat el moment angular no nul, fet que provocaria que el dron s'encallés més fàcilment amb el perfil d'alumini, augmentant el fregament i tenint un impacte negatiu en el funcionament d'aquest.

El mateix passa en els drons reals, com per exemple els quadrotors: en aquests drons s'ha de tenir en compte el mateix efecte i procurar que el moment angular total sigui nul (excepte quan es busqui precisament aquesta rotació), i per fer-ho s'han de configurar els quatre motors de manera que rotin en sentits oposats dos a dos.

2.1.4 Hèlices

Finalment, l'últim element mecànic de la maqueta són les hèlices que es munten als dos motors del dron. Les hèlices disponibles al mercat per a drons amb finalitat principalment recreatives solen estar fabricades amb un tipus de plàstic i les seves característiques més rellevants (principalment la llargada de les aspes de punta a punta i la inclinació o *pitch* que tenen aquestes) es solen codificar amb un número de 4 dígits de la següent manera: donat, per exemple, el codi 5045, tindríem que els dos primers dígits corresponen al diàmetre de les hèlices en polzades (en aquest cas 5.0 *in*, que equivalen a 12,70 cm) i els dos últims dígits representen la inclinació de les hèlices en forma de la distància que es recorreria si es donés una volta sencera d'aquestes (com si es tractés d'un cargol, en aquest cas amb un pas de 4.5 *in* o 11,43 cm). A vegades la codificació de les hèlices també incorpora informació sobre el sentit de gir pel qual estan dissenyades, tot i que no és una pràctica gaire estandarditzada donat que normalment es venen en parelles que n'inclouen una en cada sentit —tal com s'ha comentat, en la majoria dels casos es voldran tants motors (i per tant hèlices) girant en un sentit com en l'altre.

Igual com passava amb els motors, la teoria física que hi ha darrere del funcionament de les hèlices també es pot treballar amb un elevat nivell de complexitat, però a tall de referència en el nostre cas és suficient conèixer qualitativament quines característiques de les hèlices poden maximitzar la potència pel que fa a la força d'empenyiment (que no necessàriament l'eficiència, ja que com s'ha comentat no és una limitació en el nostre cas).

Així doncs, a grans trets, és important entendre que al final la força d'aixecament es tracta d'una força de reacció que s'exerceix sobre les hèlices quan aquestes, per acció del motor que les fa rotar, desplacen l'aire del seu voltant. Aquesta força, doncs, serà proporcional a la variació del moment que tingui l'aire, i per tant donada una mateixa velocitat de rotació la manera de maximitzar la variació del moment és desplaçant la quantitat més gran d'aire possible, fet que s'aconsegueix amb unes hèlices més grans i amb més inclinació. Cal ser conscients que unes hèlices més grans i amb més inclinació requereixen també més energia per a girar, però un cop més no és un paràmetre que ens limiti, de manera que es va acabar optant per les hèlices amb més diàmetre i inclinació de les que es van provar, concretament unes hèlices 5045.

Com a detall interessant a comentar, les hèlices es munten als motors per mitjà d'uns cargols femella que les subjecten a lloc. Aquests cargols estan subjectes a vibracions i fregaments durant el funcionament normal del dron que podrien arribar a afluixar-los, arribant a poder ser un perill en cas que les hèlices sortissin propulsades. Aquest fet s'agreuja quan el sentit de rotació del motor es correspon al sentit en què s'afluixen les femelles, la qual cosa no sempre es pot evitar si el sentit del cargol que incorpora el motor és el mateix pels dos (ja que, recordem un cop més, els dos motors han de girar en sentits contraris). Per evitar aquesta situació es va decidir utilitzar cargols autoblocants, que tenen la particularitat de ser especialment resistents a aquestes vibracions un cop s'han collat, de manera que sense importar el sentit de gir del motor es redueix el perill d'afluixament.

2.2 Electrònica

2.2.1 Processador

L'element principal de la part electrònica del dron és el processador, que per extensió acabarem anomenant "controlador de vol" per la tasca de coordinació dels recursos que acabarà fent. Aquest element actua com el "cervell" del dron, ja que es munta sobre el dron mateix i és l'encarregat de gestionar les entrades que rep (dels sensors o des de l'estació terrestre) i d'interpretar-les per a generar les sortides necessàries per al sistema (ordres pels actuadors i dades de *feedback* cap a l'estació).

Com és obvi, és necessari que la capacitat de processament del controlador sigui prou elevada per poder realitzar tots els càlculs necessaris a una freqüència prou alta que no comprometi el correcte funcionament de les estructures de control que haurà d'implantar per a governar el dron. Un altre requeriment del processador és que ha d'incorporar el *hardware* necessari per a dur a terme les comunicacions comentades amb el dron, per la qual cosa ha de tenir, entre d'altres, alguns ports on connectar els sensors i els connectors dels *drivers* dels motors, així com un bus o mitjà de comunicació amb l'estació terrestre.

Sent conscients d'aquests requeriments, i també pensant en escollir una opció que fos accessible i còmode d'utilitzar per als usuaris de la plataforma docent que es buscava crear, la plataforma escollida per a actuar com a processador del dron va ser una placa *arduino*. Aquesta plataforma resulta ser molt popular i de fet està especialment dissenyada per al prototipatge ràpid associat amb els projectes TIC, i és una eina molt utilitzada en educació, ja que ofereix un gran rang de possibilitats amb un cost raonable i molta flexibilitat, a més de ser *hardware* obert i tenir una gran comunitat molt activa darrere que hi dóna suport i facilita l'entrada als usuaris sense experiència. El projecte *Arduino* i la porta d'entrada a la comunitat que hi dóna suport es troba a [**arduino**].

La placa escollida, *Arduino UNO* (que es pot observar a la figura 2.8, té un seguit de

Figura 2.8: Placa *Arduino* UNO utilitzada per al nostre projecte. Font: *arduino.cc*.

característiques que a priori sembla que la fan una opció vàlida per a ser el processador amb els requeriments que s'ha comentat que eren necessaris: º

1) Microcontrolador *atmega328p* ([**atmega**]) de 8 bits amb una freqüència de *clock* de 16,00 MHz
2) Port sèrie amb un processador dedicat que fa de pont amb un connector USB tipus B
3) Alimentació a 5,00 V disponible directament des del connector USB
4) 6 ports *built-in* de sortida PWM
5) 14 ports I/O digitals

Si bé aquestes característiques ja possibiliten l'ús de la placa *Arduino* com a controlador de vol per al nostre dron simplificat, cal indicar que en un projecte orientat al disseny d'un producte final del qual se'n busqués el màxim rendiment i eficiència, també pel que fa al cost i al consum, possiblement aquesta no seria l'opció escollida. Per norma general, una placa electrònica o PCB (de l'anglès *Printed Circuit Board*) dissenyada a mida, que incorpori només els elements estrictament necessaris donats els requeriments d'un projecte determinat, sempre serà més barata (si es contempla el preu individual), tindrà un consum menor (el just per a complir les seves funcions) i podrà un pes i una mida més reduïts. Així i tot, com ja s'ha comentat, en el nostre cas l'accessibilitat que permet l'*Arduino* va ser que s'acabés optant per aquesta opció.

Una raó que també va fer decantar la balança a favor d'utilitzar una placa *Arduino* és l'existència d'un IDE especialment dissenyat per treballar amb aquestes plataformes. Aquest IDE

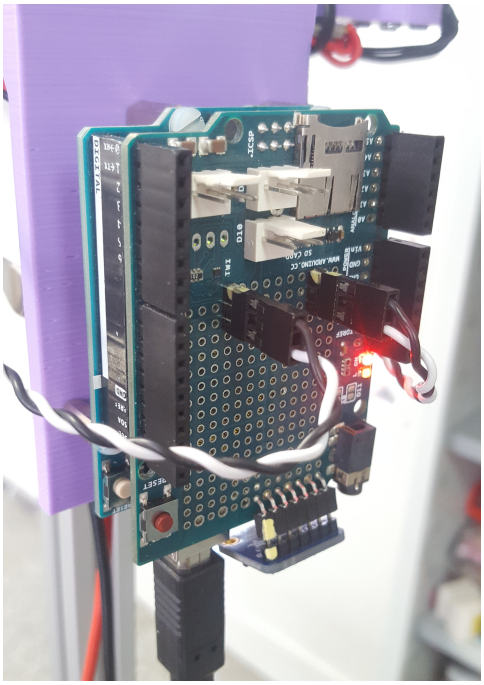

Figura 2.9: Placa *Arduino* muntada sobre el dron amb la *shield* incorporada.

(de l'anglès *Integrated Development Environment*) incorpora diverses facilitats en la programació d'aquestes plaques, i permet escriure programes en el llenguatge de programació C++ que incorporin funcions i llibreries especialitzades per a treballar amb les plataformes *arduino*. Aquest fet que facilita enormement la tasca de programar i modificar el codi que s'acaba injectant al processador del dron ha estat doncs encara un motiu més a l'hora d'escollir aquesta opció, sempre amb la intenció que l'eina acadèmica resultant d'aquest projecte sigui realment útil i flexible.

Finalment, és interessant comentar que a la placa *Arduino* UNO utilitzada en el dron s'hi ha acoblat un mòdul *hardware*, també anomenats *shield*, que incorpora una zona de pins verges per a soldar-hi components electrònics per tal d'afegir-hi els diferents connectors que utilitzats per a connectar la resta d'elements electrònics amb el processador. Aquest tipus de pràctiques són molt corrents en l'ambient de prototipatge ràpid i modular que persegueix el projecte *arduino*. A la figura 2.9 es pot observar la placa *Arduino* amb la *shield* muntada un cop els connectors soldats i connectats.

2.2.2 ESCs

Els controladors o *drivers* dels motors utilitzats pels drons sovint s'anomenen ESCs (de l'anglès *Electronic Speed Control*) per la funció tan especialitzada que realitzen de control de la velocitat d'aquests. Aquests controladors incorporen els components elèctrics i electrònics necessaris per realitzar, com a mínim, les següents funcions:

1) Convertir l'alimentació de DC a AC per a alimentar el motor que controlen

2) Regular la freqüència d'alternació de la tensió per a ajustar la velocitat de rotació del motor d'acord amb el senyal d'entrada rebut

3) Permetre la comunicació amb el controlador mitjançant un cert protocol per tal de poder ajustar la velocitat de rotació del motor

Tenint en compte que els ESCs són components que han de proporcionar l'alimentació a uns actuadors amb un consum relativament elevat com són els motors (comparat amb altres components electrònics de consum de l'ordre dels mA, com ara LEDs o brunzidors), és important tenir en compte els requeriments energètics d'aquests últims a l'hora de seleccionar el *driver* que millor pugui proporcionar l'alimentació necessària pel motor en tot el seu rang de funcionament. En el nostre cas, els dos models de motors que s'han utilitzat requereixen un ESC de 30,00 A (informació proporcionada amb les especificacions dels models concrets) i poden funcionar a una tensió de fins a 14,80 V (equivalent a quatre cèl·les *Lipo* connectades en sèrie), de manera que es va escollir uns *driver* d'aquestes característiques.

Pel que fa als mitjans de comunicació utilitzats pels controladors de velocitat, existeixen diversos protocols, que principalment es distingeixen per si són analògics o digitals. El principal protocol de comunicació analògic, que és el que s'ha utilitzat, és el basat en polsos PWM , que utilitza polsos d'una freqüència marcada de 50,00 Hz amb un rendiment de cicle variable per a representar els diferents valors d'entrada que pot llegir el controlador —es considera el valor mínim de velocitat per a polsos de 1,00 ms i el màxim per a polsos de 2,00 ms. Per a llegir correctament aquests valors i ajustar d'alguna manera la sincronització entre els controladors dels motors i el controlador de vol, els protocols analògics de comunicació entre aquests incorporen un procediment de calibració durant el qual es defineixen els límits superior i inferior del senyal PWM utilitzat, en el nostre cas.

Altres protocols, com els de tipus digital, utilitzen estratègies semblants per a transmetre informació, permetent normalment majors velocitats de comunicació, retards menors en la transmissió o la no necessitat de calibrar els controladors, entre d'altres, a canvi d'electrònica més complexa i implementacions en *software* dels protocols que, sovint, són privatius o d'una complexitat molt major a la d'un senyal PWM. És per aquest motiu que tot i els avantatges que poden comportar protocols més nous i potents com per exemple *OneShot MulitShot* o *DShot* (protocols digitals utilitzats per alguns populars controladors disponibles al mercat), s'ha acabat optant per un protocol més simple i obert com és el protocol PWM per simplicitat (un aspecte positiu si es té en compte que es busca que la maqueta desenvolupada pugui ser fàcilment accessible i fins i tot modificada) i facilitat d'implementació —molts microcontroladors, com l'*atmega328p* escollit per al nostre dron, incorporen la funcionalitat de generar senyals PWM sense necessitat de programar-les expressament.

Una característica interessant dels ESCs escollits per al dron, concretament el model *Diatone 302XD* (que es pot observar a la figura 2.11), es que incorporen una funcionalitat anomenada "fre actiu", en anglès *active brake*, que fa que la frenada dels motors quan baixa la demanda es faci expressament, és a dir, que en comptes d'esperar que els motors perdin velocitat pel fregament quan es redueix la velocitat comandada, s'utilitza el moviment dels propis motor per a generar energia elèctrica, d'aquesta manera reduïnt abans l'energia cinètica que té el motor. Si bé això permet uns temps de reacció dels motors més curts en la frenada, un efecte col·lateral és que durant aquest procés es generen uns pics de tensió que poden crear corrents en sentit contrari (del controlador a la font d'alimentació, en aquest cas). Es va detectar que quan s'atura el motor mentre aquest està girant a velocitats relativament elevades (a partir d'aproximadament el 80 % de la capacitat màxima) això arriba a ser problemàtic, ja que la mateixa font es protegeix d'aquest corrent induït bloquejant-se. És per aquest motiu que es

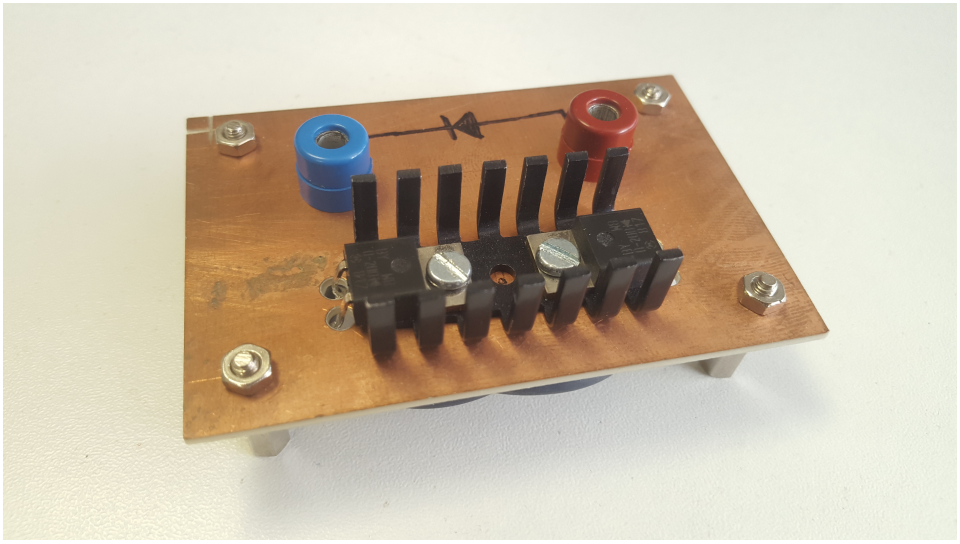

Figura 2.10: Placa amb els díodes de protecció de la font i els dissipadors de calor corresponents.

van afegir uns díodes de protecció a l'entrada de la font, (a la figura 2.10 es pot observar una fotografia de la placa que els inclou), a més de tenir en compte que és preferible evitar les frenades brusques a l'hora de programar el controlador de vol.

Finalment, és interessant comentar que aquests controladors, especialment dissenyats pel seu ús en drons, sovint incorporen funcionalitats extres que en general estan dissenyades per a fer més segur l'ús d'aquests. Algunes d'aquestes funcions de seguretat que incorporen els controladors escollits són, per exemple, el bloqueig dels motors des que s'alimenten fins que no es rep un senyal PWM amb el valor mínim durant un cert període de temps per evitar que els motors comencin a girar de manera inesperada quan s'encén el dron, o diferents tons auditius per a comunicar situacions diverses (calibració en procés, encesa dels motors, motors encesos però no operats durant l'últim minut per tal de poder-lo localitzar en cas de caiguda i pèrdua d'un dron real, etc.).

2.2.3 Alimentació

Tot i que s'han realitzat projectes que exploren alternatives pel que fa al mitjà d'alimentació dels drons (actualment al mercat es poden trobar els anomenats *tethered drones*, que es tracten de drons alimentats per cable), aquest ha estat per excel·lencia les bateries que aquests mateixos incorporen. El principal avantatge que proporcionen les bateries és que permeten al dron no tenir un rang de moviment limitat, i per tant volar sense més restriccions que l'abast de la comunicació ràdio amb l'estació de control o la capacitat de la bateria mateixa.

Al mateix temps, però, l'ús de bateries comporta principalment dos inconvenients importants. Per una banda la bateria té un pes determinat que s'afegeix al del dron, requerint més potència als motors per aconseguir els mateixos resultats que si no es portés la bateria. A més, per definició l'autonomia de vol del dron està limitada per la duració de la càrrega de la bateria, la capacitat màxima de la qual està relacionada amb la mida d'aquesta (per tant tornem al primer inconvenient que s'exposava).

En el cas dels drons reals sovint s'accepten aquests inconvenients a canvi de tenir la llibertat de moviment que es comentava al principi d'aquest apartat, però en el nostre cas, que parteix

Figura 2.11: El controlador de motors *Diatone 302XD* utilitzat en la maqueta. Font: *Diatone*.

d'un dron amb els graus de llibertat limitats, resulta que no té tant sentit optar per l'opció de les bateries com a font d'alimentació, ja que de tota manera no podríem tenir llibertat de moviment total. És per aquest motiu que com a mètode d'alimentació de la maqueta es va escollir l'alimentació per cable, connectant els controladors dels motors amb una font d'alimentació (i la plataforma *Arduino* a través de l'alimentació del mateix cable USB que s'hi pot connectar.

Una conseqüència d'utilitzar alimentació per cable és que d'alguna forma el pes del dron passa a ser variable, ja que el pes del cable (que s'exerceix sobre el dron en si) depèn de l'alçada. Això pot tenir certs efectes que pel que fa a les estructures de control necessàries per a governar el vol del dron, que es comentaran més endavant.

2.2.4 Sensors

Un aspecte essencial quan es tracta d'estructures de control amb llaç tancat (és a dir, amb *feedback* sobre l'estat de la planta com es pretén tenir en el nostre projecte) són els sensors que ha de tenir el sistema per obtenir dades sobre l'operació dels actuadors i l'efecte que estan tenint sobre la planta. Tot i que els drons del mercat poden incorporar diferents tipus de sensors en funció de les aplicacions que se'ls vulgui donar, alguns dels més usuals són:

1) IMU (*Inertial Measurement Unit*)
    a) Acceleròmetre: permet determinar les acceleracions en els tres eixos
    b) Giroscopi: permet determinar les velocitats de rotació en els tres eixos

c) Magnetòmetre: permet determinar la força dels camps magnètics en els tres eixos

2) GPS: permet determinar la posició (especialment latitud i longitud) a partir del sistema de geoposicionament GPS

3) Baròmetre: permet determinar l'alçada a partir de la pressió atmosfèrica

4) Sensor d'ultrasons: permet detectar la distància a partir de la reflexió d'ultrasons als objectes propers al dron

Tot i la varietat de sensors que un dron pot incorporar, a priori es va considerar que pel nostre projecte, donat que al final només governar el dron amb una consigna de posició en l'eix vertical (recordem, passem de tenir sis graus de llibertat a tenir-ne només un), un sensor que directament llegeixi aquesta magnitud seria suficient. Així, es va acabar optant per incorporar al dron un sensor de distància basat també en el "temps de vol" però en aquest cas de pulsos làser (i no ultrasons, com en el cas d'un típic sensor de drons reals que es comentava a dalt).

Finalment el sensor seleccionat va ser un sensor LIDAR (de l'anglès *Light Detection and Ranging*) de la marca *ST*, concretament el model *VL53L0X* ([**vl53l0x**]).

El funcionament bàsic d'aquest sensor es basa en la tècnica coneguda com a *time-of-flight*, que consisteix a emetre polsos (en aquest cas, polsos làser) i mesurar el temps que es tarda a detectar-ne la reflexió. En funció d'aquest temps es pot mesurar a quina distància es troben els objectes més propers al dron —com més lluny es trobin, més es tardarà a rebre la reflexió dels polsos. Cal tenir en compte que la precisió i efectivitat del sensor depèn en gran manera de les condicions d'utilització, com per exemple la disposició dels objectes en l'espai al seu voltant, la distància respecte al sensor d'aquests i el material, textura o fins i tot color d'aquests objectes.

A l'hora d'utilitzar aquests sensors s'han hagut de tenir algunes consideracions especials. Per exemple, s'ha procurat de mantenir lliure d'objectes que podrien generar reflexions indesitjades un con sòlid de 25,00° davant el receptor de polsos, d'acord amb les especificacions de l'integrat, ja que aquest és l'angle de visió que té el sensor tal com es pot observar a la figura 2.12.

Una altra consideració que s'ha hagut de tenir en compte és que la precisió del sensor es degrada amb la distància: d'acord amb les especificacions del fabricant, a més distància (fins a un màxim de 2,00 m, el rang màxim del sensor) més precisió té el sensor, tal com es pot observar a la figura 2.13. Això implica que el funcionament del dron serà pitjor a més alçada, ja que en aquestes condicions el senyal obtingut té més soroll.

Aquest comportament especificat pel fabricant del sensor resulta complir-se a la realitat, tal com es pot observar a la figura 2.14, que correspon a lectures extretes del sensor. Es pot observar clarament com les dades fluctuen més a mesura que augmenta la distància entre el sensor i els objectes que té al seu davant (en aquest cas, el terra).

El sensor disposa d'una API o interfície de programació d'aplicacions (en anglès *Applicacion Programming Interface*) que permet ajustar-ne la configuració d'acord amb alguns perfils de funcionament predefinits. En funció de la combinació d'aquests modes i perfils de funcionament es configuren algunes magnituds des del mateix sensor per a tenir els resultats esperats, com ara el llindar mínim per a considerar una reflexió com a vàlida o la durada dels polsos làser, per exemple (l'API precisament busca estalviar a l'usuari haver d'aprofundir fins a aquest nivell en el funcionament del sensor). Aquests perfils i modes de funcionament són els següents:

1) Modes de mesura:

a) Mesura única: el sensor realitza una única lectura quan es crida l'API

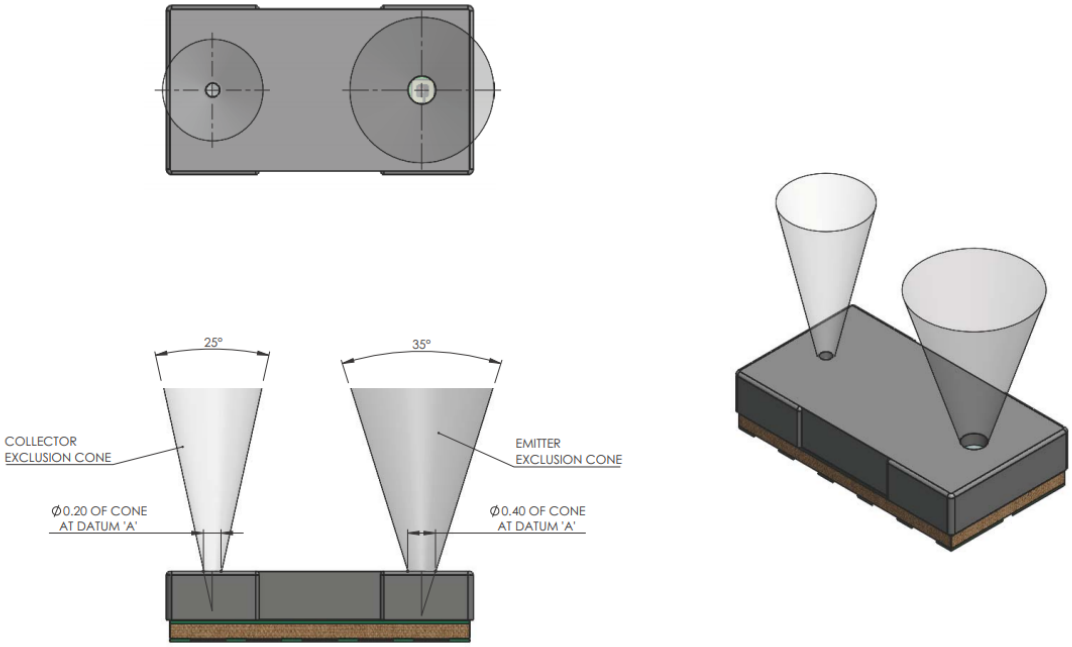

Figura 2.12: Con sòlid que representa l'angle de visió del sensor *VL53L0X*. Font: *ST*.

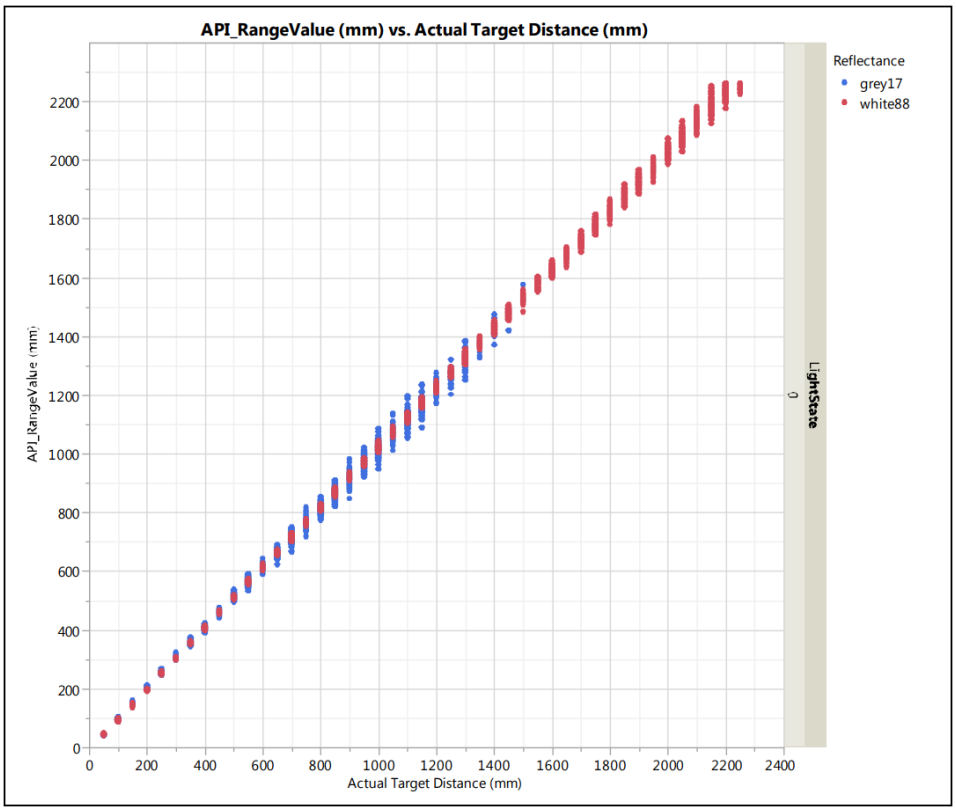

Figura 2.13: Precisió del sensor en funció de la distància. Font: *ST*.

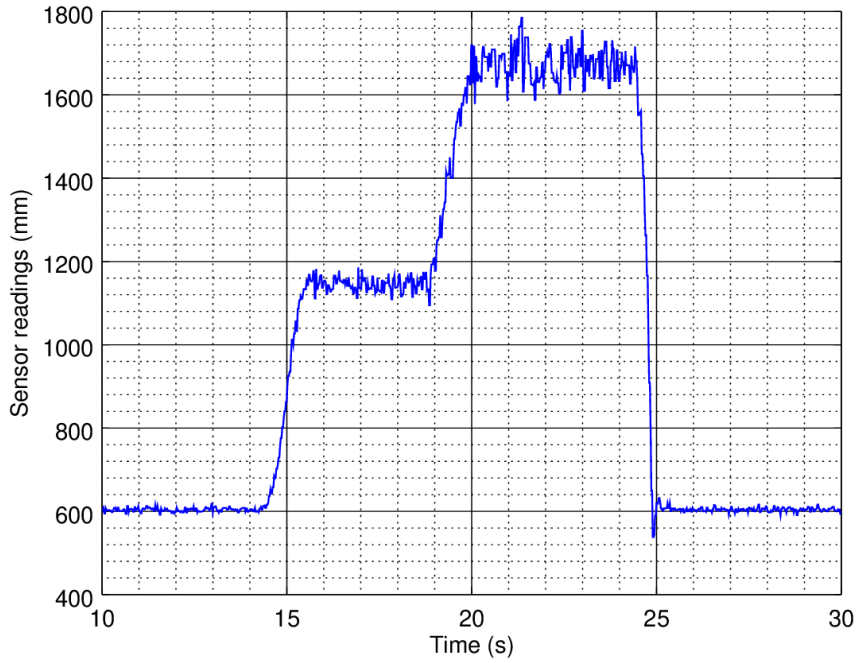

Figura 2.14: Lectures de distància extretes del sensor que mostren la variació del soroll en funció de la distància del sensor als objectes.

b) Mesura contínua: el sensor realitza lectures tan ràpid com li és possible, iniciant-ne una cada vegada que acaba l'anterior

c) Mesura temporitzada: el sensor realitza lectures amb un període entre mesura especificat per l'usuari

2) Perfils de mesura:

a) Mode per defecte

b) Alta velocitat

c) Alta precisió

d) Larg abast

Pel que fa al mode de funcionament, en el nostre cas s'ha optat per escollir el mode de mesura contínua, ja que d'aquesta manera el període entre mostres es minimitza, i en el nostre cas això ens pot beneficiar, ja que permet una velocitat de mostreig màxima, cosa preferible si es busca un control capaç de governar correctament la planta. Al mateix temps, s'han seleccionat els perfils d'alta velocitat i llarg abast per poder treballar a una distància de fins a 2,00 m i amb un període entre mostres mínim. La freqüència de mostreig màxima que es pot assolir amb aquest sensor és de 50,00 Hz, ja que en el mode d'alta velocitat (que utilitza el menor temps possible entre mostres) aquest període és de 20,00 ms.

És interessant comentar que a causa de la degradació de la precisió que experimenta el sensor a mesura que augmenta la distància als objectes detectats, i en general per tal de millorar la qualitat de les dades obtingudes, una opció a considerar és la de realitzar un filtratge de les

lectures realitzades pel sensor. Aquest sensor normalment és de tipus pas baix, ja que es pretén eliminar el soroll d'alta freqüència (en comparació a les variacions en la distància). Així doncs, un dels filtres més senzills que es pot implementar és un filtre de mitjana mòbil, que consisteix a considerar les dades del senyal filtrat com la mitjana d'un subconjunt de les dades començant per les immediatament anteriors a la mostra actual. Així, tindríem que una implementació d'un filtre d'aquest tipus a temps real tindria una expressió com la que segueix:

$$\bar{x}(n) = \frac{1}{m} \sum_{i=0}^{m-1} x(n-i)$$

On $x(n)$ és el senyal a filtrar, m la mida de la finestra mòbil (que indica de quants valors anteriors es fa la mitjana) i $\bar{x}(n)$ el senyal filtrat. Cal tenir en compte que aquest filtre, si bé permet obtenir uns resultats prou bons pel que fa a la reducció del soroll, també introdueix un cert retard temporal al senyal, ja que l'efecte de les noves lectures (en cas d'un canvi sobtat, per exemple) no té un impacte important en el senyal filtrat fins al cap d'unes quantes mostres. Això serà un problema a l'hora d'utilitzar aquest senyal per al nostre control, com es discutirà més endavant.

Una altra alternativa pel que fa als possibles filtres que es podrien utilitzar és l'anomenat filtre de mediana, que utilitza una tècnica semblant al filtre de la mitjana mòbil (que per tant té els mateixos inconvenients, com ara el retard) però calculant la mediana del subconjunt de dades de la finestra mòbil, d'acord amb una expressió com la següent:

$$x'(n) = med([x(n), x(n-1), x(n-2), \ldots, x(n-m-1)])$$

On $x(n)$ és el senyal a filtrar, m la mida de la finestra mòbil i $x'(n)$ el senyal filtrat.

Després de fer proves amb els dos filtres, es va acabar optant per utilitzar el primer, ja que es va determinar que duia a terme una millor tasca a l'hora d'eliminar soroll d'alta freqüència que podria afectar més negativament el funcionament del control.

3 Desenvolupament del programari

En aquest capítol es descriuen les eines *software* desenvolupades per a poder controlar el funcionament del dron i fer la recollida i tractament de dades. A més, aquestes eines s'han pensat de manera que permetin també modificar alguns paràmetres del funcionament del dron "en calent", mentre aquest està encès, per fer possible que el procés d'assaig i error propi de l'experimentació amb la maqueta sigui més àgils.

A grans trets, tenim dos grans blocs pel que fa al codi desenvolupat per aquest projecte. En primer lloc, tenim el codi corresponent al dron, que és un programa dividit en blocs i programant en el llenguatge de programació C++ que s'injecta a la placa *Arduino* que incorpora el dron a tall de processador. L'estructura de mòduls d'aquest programa, juntament amb un breu resum del seu contingut, és el següent:

1) `Drone.ino`: és el programa principal, i és el que realitza les crides a la resta de mòduls per a implementar la màquina d'estats i gestiona la comunicació amb l'estació terrestre.
    a) `Controller.h`: inclou la classe `Controller`, que ofereix el tipus d'estructura de dades utilitzat en tot el programa així com funcions per a actualitzar i en general manipular aquestes dades d'acord amb l'estat del dron. Inclou també la implementació del controlador PID que s'ha utilitzat per a aquest projecte, i les funcions relacionades.
    b) `Sensor.h`: inclou la classe `Sensor`, que ofereix funcions per a configurar el sensor de distància i per a obtenir-ne lectures.
    c) `Motor.h`: inclou la classe `Motor`, que ofereix funcions per a controlar el rendiment de cicle del senyal que governa els motors (i per tant, la seva velocitat).

Pel que fa al codi corresponent al programa de l'estació terrestre, aplicació que s'executa en un ordinador (durant el projecte s'ha treballat concretament amb el sistema operatiu *Ubuntu 16.04*), també podem diferenciar un seguit de mòduls que conformen el conjunt del programa, escrit en el llenguatge de programació `Python` (concretament la versió `3.5` del llenguatge):

1) `Station.py`: programa principal que inclou la classe `Station`, que actua com a pont per a les subclasses `GUI` i `SerialInterface` per a integrar totes les parts del programa i coordinar la comunicació amb el dron amb les dades mostrades per pantalla
    a) `GUI.py`: inclou la classe `GUI`, que gestiona tots els elements de la interfície gràfica del programa i realitza les crides als corresponents submòduls per a generar els diferents marcs[1] que formen la finestra principal del programa

[1] Entenem com a "marcs" als elements en què està dividida una finestra d'un programa informàtic, que al seu torn poden incloure d'altres elements (alguns noms comuns per aquest tipus d'estructures en anglès són *frame*, *division*, *box*, etc.). Aquest tipus d'estructures que permeten organitzar més fàcilment les finestres d'un programa són una pràctica molt estesa en diverses plataformes o llibreries d'interfícies d'usuari

i) `SetupFrame.py`: inclou la classe `SetupFrame`, que es correspon amb el marc amb les opcions de configuració del dron

ii) `PlotFrame.py`: inclou la classe `PlotFrame`, que es correspon amb el marc de les gràfiques en temps real

iii) `CheckFrame.py`: inclou la classe `CheckFrame`, que es correspon amb el marc amb les opcions de visualització de les gràfiques

iv) `ButtonFrame.py`: inclou la classe `ButtonFrame`, que es correspon amb el marc amb els botons de control del dron

v) `LogoFrame.py`: inclou la classe `LogoFrame`, que es correspon amb el marc amb el logo de l'EPSEM

vi) `StateFrame.py`: inclou la classe `StateFrame`, que es correspon amb el marc amb la informació a temps real de l'estat del dron

b) `SerialInterface.py`: inclou la classe `SerialInterface`, que gestiona la comunicació amb el port sèrie

i) `Message.py`: inclou la classe `Message` i les seves subclasses, que corresponen als blocs de transmissió del protocol de comunicació entre l'estació i el dron

A continuació s'explica amb més detall el funcionament dels diferents mòduls que conformen les eines *software* que s'han desenvolupat per aquest projecte. De manera semblant a com s'han llistat en aquesta introducció del capítol, primer es parla del programa injectat al dron i després del programa que actua com a estació terrestre del nostre sistema.

Com a norma general, es procurarà de no referenciar parts del codi directament, sinó que s'intentarà explicar el seu funcionament a més alt nivell per entendre com s'estructuren i s'interrelacionen totes les peces que formen aquestes eines. Així doncs, en comptes d'enganxar grans fragments de codi es procurarà de fer èmfasis en els algorismes, estructures i relacions entre blocs per tal de proporcionar unes idees més clares i concises, posant èmfasis en les decisions de disseny preses durant el desenvolupament. El codi dels programes en si està adjuntat a aquesta memòria per a la seva consulta.

3.1 Dron

Tal com s'ha comentat abans, el codi desenvolupat per al dron en si s'ha escrit en el llenguatge de programació C++ , concretament la versió modificada que utilitza l'IDE d'*arduino*. Aquesta versió modificada ofereix, entre d'altres, facilitats per a treballar amb les entrades i sortides de la placa, funcions especialitzades per a comunicar-se més còmodament amb el port sèrie i fins i tot llibreries que implementen protocols de comunicació com ara el protocol I^2C, que és l'utilitzat pel sensor de distància del nostre dron (tot i que en aquest cas s'ha utilitzat directament una llibreria especial per a aquest sensor, que ofereix les funcions de l'API d'aquest sense necessitat de preocupar-se explícitament pels detalls d'implementació del protocol I^2C.

Així doncs, aprofitant també les facilitats que ofereix aquest IDE per a la compilació i programació de la placa, es va optar per dividir el programa que es carregaria al dron en diferents mòduls per claredat i millor organització del codi. Per una banda, tenim el mòdul principal, l'arxiu `Drone.ino`, que té l'extensió pròpia dels fitxers de projectes de l'IDE d'*Arduino* , i per l'altra, tres arxius amb extensió `.h` que s'inclouen a l'arxiu principal en el moment de la compilació. A la figura 3.1 podem observar un diagrama que pretén il·lustrar les relacions entre

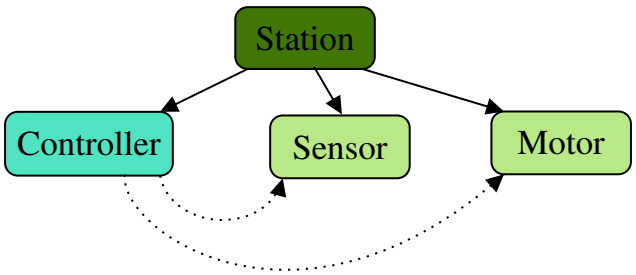

Figura 3.1: Diagrama de mòduls del programa del dron.

els diferents mòduls que formen aquest programa, que s'explicaran a continuació. Noteu que per aquest diagrama no s'han tingut en compte les extensions dels noms dels mòduls, i de fet s'ha assumit directament que cada mòdul es correspon amb una classe de C++ si bé això no és exactament així, ja que el mòdul principal senzillament actua com a punt d'unió de la resta —això sí, seguint l'estructura pròpia dels projectes *Arduino* basada en les dues funcions bàsiques `setup()` i `loop()`.

Com es pot observar, la principal via de comunicació entre el programa principal i els actuadors de la planta (en el nostre cas, els motors) és a través del controlador, és a dir, l'objecte de la classe `Controller`. El sentit de les fletxes del diagrama de la figura 3.1 representa quin mòdul crida a l'altre, i per tant ens dóna una idea d'aquesta estructura que comentàvem.

A continuació es detallen les classes i funcions més rellevants que s'inclouen a cada mòdul per tal d'il·lustrar millor la relació entre tots ells.

3.1.1 Mòdul `Motor.h`

Aquest mòdul és una abstracció de la classe que conté, la classe `Motor`. Aquesta classe consta del següents mètodes[2]:

1) `void setup()`

   Aquesta funció configura els senyals PWM que actuen com a entrada dels controladors dels motors de la planta, i inicialitza la durada dels polsos al valor mínim.

2) `void setSpeed(int n, int val)`

   Aquesta funció configura la durada dels polsos del senyal PWM corresponent al motor `n` (que pot prendre els valors enters 1 o 2 per a representar a quin dels dos motors es fa referència) en funció del paràmetre `val`, que representa el rendiment de cicle com a percentatge del total (un valor de `val` igual a 0 es tradueix al valor mínim de durada dels polsos, i per tant al motor apagat, i un valor de 100 es tradueix a la durada màxima dels polsos, per tant a la màxima velocitat que el motor pot assolir). Addicionalment, la funció emmagatzema com a atribut de la classe l'última velocitat configurada per a cada motor i incorpora mecanismes de seguretat com interpretar com a 0 o 100 els valors fora d'aquest interval i ignorar les crides que configuren un motor inexistent (diferent de 1 o 2).

[2] "mètode" és el nom que reben les funcions pròpies d'una classe en el llenguatge C++

3) `int getSpeed(int n)`

   Aquesta funció retorna l'últim valor de velocitat configurat al motor `n` com a enter que representa el percentatge respecte a la màxima velocitat que pot assolir el motor.

Les funcions d'aquesta classe són cridades pel programa principal (concretament les funcions de `setup` i `getSpeed`) a manera de supervisor i per la classe `Controller` quan actualitza el valor de velocitat requerit d'acord amb la sortida del control que implementa el dron.

3.1.2 Mòdul `Sensor.h`

Aquest mòdul és una abstracció de la classe que conté, la classe `Sensor`. Aquesta classe consta dels següents mètodes:

1) `void setup()`

   Aquesta funció configura el sensor de distància segons les característiques de funcionament que s'han determinat òptimes, inclosos els ajustaments propis del mode de funcionament de mesura contínua i els perfils d'alta velocitat i llarg abast.

2) `int getReading()`

   Aquesta funció retorna l'última dada correcta que ha llegit el sensor.

Les funcions d'aquesta classe són cridades pel programa principal (que centralitza la configuració dels diferents mòduls del programa) i per la classe `Controller` quan realitza una lectura del sensor com a part de la rutina d'actualització de l'estat del control.

3.1.3 Mòdul `Controller.h`

Aquest modul és una abstracció de la classe que conté, la classe `Controller`. Aquesta classe inclou la definició del tipus de dada `controllerData_t`, definit de la següent manera en el codi:

```
1  typedef struct controllerData_t {
2    // size 2 bytes
3    unsigned int setPoint = 0xffff,
4    reading = 0xffff,
5    position = 0xffff;
6    //size 2 bytes
7    int error = 0xffff;
8    // size 4 bytes
9    long iState = 0xffffffff,
10   iMax = 0xffffffff,
11   iMin = 0xffffffff;
12   //size 2 bytes
13   unsigned int dState = 0xffff;
14   // size 4 bytes
15   float pGain = 0xffffffff,
16   iGain = 0xffffffff,
17   dGain = 0xffffffff;
```

```
18      float pTerm = 0xffffffff,
19      iTerm = 0xffffffff,
20      dTerm = 0xffffffff;
21      float drive = 0xffffffff;
22      // size 2 bytes
23      unsigned int command = 0xffff,
24      commandMax = 0xffff;
25      // size 4 bytes
26      unsigned long time = 0xffffffff;
27      // size 2 bytes
28      unsigned int state = 0xffff;
29  };
```

Aquesta estructura de dades és utilitzada també pel programa principal, i representa totes aquelles dades que tenen a veure amb l'estat del dron en si, ja sigui pel que fa a l'estat corresponent en el diagrama d'estats, la marca de temps o els valors d'entrada i sortida utilitzats pel control. Més endavant es podrà veure com aquesta estructura també és la utilitzada per a la transmissió de dades entre l'estació i el dron i viceversa.

A més d'aquesta definició de tipus, aquesta classe també consta dels següents mètodes rellevants:

1) `void resetData()`

   Aquesta funció reinicialitza els valors de l'atribut propi de la classe del tipus `controllerData_t`, anomenat `data`, utilitzat per a mantenir l'estat del control.

2) `void setup()`

   Aquesta funció senzillament crida la funió `resetData()`, existeix per a mantenir l'ortogonalitat amb el tractament de la resta de mòduls des del programa principal.

3) `void setData(controllerData_t newData)`

   Aquesta funció sobreescriu el valor de l'atribut propi `data` amb els valors del paràmetre del tipus `controllerData_t` passat com a argument.

4) `controllerData_t getData()`

   Aquesta funció retorna el valor de l'atribut d'estat `data`.

5) `void readSensor()`

   Aquesta funció crida el mètode `getReading()` del mòdul `Sensor` per a obtenir una nova mostra, i aplica el filtratge de les dades per acabar modificant el valor dels camps `reading` i `position` de l'atribut d'estat `data`.

6) `float updatePID(float position)`

   Aquesta funció implementa el controlador PID que s'ha escollit per a avaluar el funcionament de la maqueta i retorna el valor calculat per al senyal de control de la planta.

7) `void update(bool manual, unsigned int value)`

   Aquesta funció actualitza el valor del senyal de control dels motors cridant la funció de la classe `Motor` corresponent amb el valor desitjat com a paràmetre, que depèn del mode de funcionament actiu en aquell moment (manual o automàtic).

8) `void updateAuto()`

   Crida la funció `update()` en mode automàtic amb el valor d'alçada desitjat (el punt de referència per al nostre control).

9) `void updateManual()`

   Crida la funció `update()` en mode manual amb el valor d'entrada dels motors desitjat (introduït per l'usuari).

Les funcions d'aquesta classe actuen d'alguna manera com a API per al programa principal que li permet interactuar amb el control del dron en si. Les crides a aquestes funcions formen part de la rutina periòdica que realitza el programa principal, que es detalla al següent apartat.

Observant el codi es pot apreciar que alguns dels càlculs realitzats en les funcions del mòdul, especialment la funció encarregada d'implementar el controlador PID (`updatePID`), es fan sobre variables del tipus `float`. En comparació amb una altra implementació que busqués la màxima velocitat i eficiència en el càlcul utilitzant estratègies com ara l'ús dels tipus `int` i diferents operacions matemàtiques per reduir el nombre d'instruccions del processador per a fer els càlculs, és innegable que la implementació que s'ha fet resulta més lenta i ineficient.

El motiu pel qual això no s'ha considerat un fet crític és que, en comparació amb la freqüència del programa principal (que acaba consistint en un bucle repetitiu), resulta que la suma dels temps de processament del nostre programa no arriba a ser un inconvenient, ja que la velocitat màxima està limitada per la freqüència màxima del sensor de distància (que com s'ha comentat és de 50,00 Hz), i el temps de càlcul no arriba ni tan sols a la meitat del període de mostreig (fins i tot tenint en compte les operacions relacionades amb el port sèrie). Com és obvi, donades unes diferents condicions com per exemple l'ús d'un sensor més ràpid, s'hauria de tenir en compte aquest aspecte i utilitzar estratègies per a l'optimització dels temps de càlculs per evitar que acabessin sent problema.

3.1.4 Mòdul `Drone`

El mòdul `Drone.ino` és, de fet, el programa principal que implementa les funcions `setup()` i `loop()` que apareixen en tots els projectes d'*Arduino* programats a través del seu IDE.

De fet, en el nostre programa la funció que marca en quin moment s'ha d'executar una determinada seqüència d'accions és una rutina d'interrupció que genera el temporitzador 2 del mateix microcontrolador, l'*atmega328p*. Aquesta rutina d'interrupció corresponent, configurada perquè tingui lloc cada 10,00 ms, incrementa un sumador que comptabilitza el nombre de milisegons des de l'última interrupció, permetent així determinar a la funció `loop()` quan és el moment d'executar la rutina de nou (quan la quantitat de milisegons transcorreguts és la correcta). Amb aquest enfocament, s'aconsegueix que la rutina d'interrupció del nostre programa no sigui bloquejant, una bona pràctica que pot evitar errors difícils de detectar al codi. Aquest esquema es pot observar en el següent fragment de codi, que també inclou crides a la funció pròpia d'*Arduino* `digitalWrite` que permeten observar que efectivament la rutina principal del programa s'executa a la freqüència desitjada per mijà d'un pin extern de la placa, que està a nivell alt durant l'execució de la interrupció i a nivell baix durant el període intermedi.

```
1  ISR(TIMER2_COMPA_vect) {
2    static int times10ms = 0;
3    times10ms++;
```

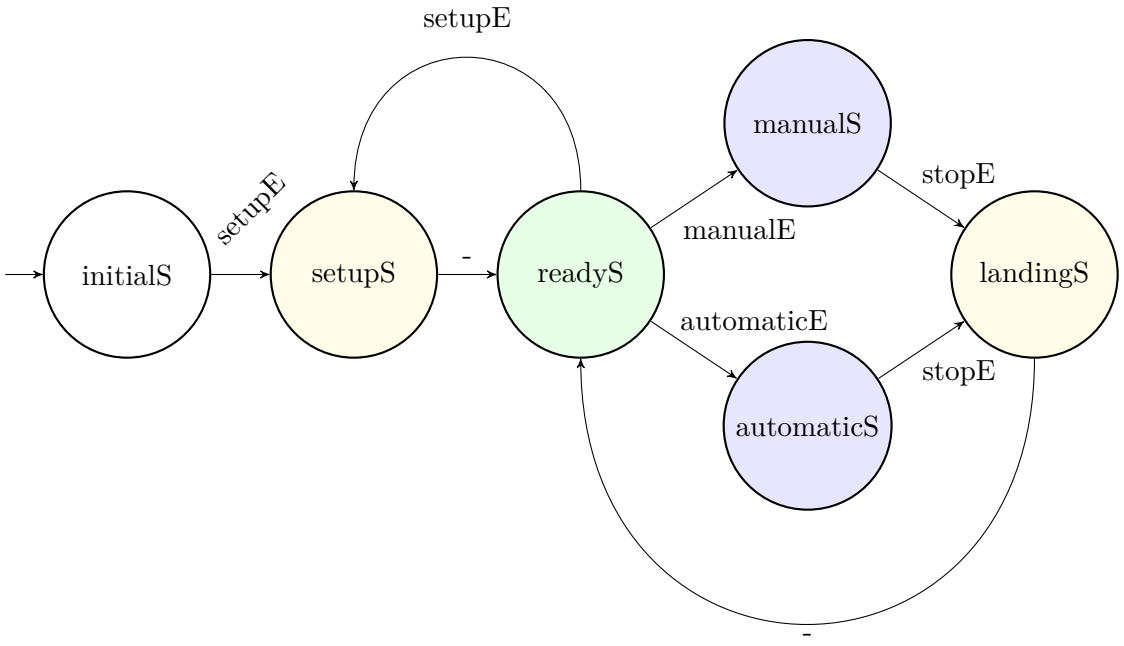

Figura 3.2: Diagrama d'estats del dron.

```
4     if (times10ms == 2) {
5       interrupt = true;
6       times10ms = 0;
7     }
8   }
9
10  void loop() {
11    if (interrupt == true) {
12      digitalWrite(pin, HIGH);
13      interrupt = false;
14      routine();
15      digitalWrite(pin, LOW);
16    }
17  }
```

Aquesta rutina principal de què s'ha parlat duu a terme diverses accions una darrere l'altra que contribueixen a realitzar tots els càlculs necessaris per determinar el nou estat del sistema després de cada iteració. Aquestes diverses es corresponen de fet amb les diverses funcions que s'han implementat al programa principal, de manera que descrivint aquestes funcions per ordre es pot tenir una idea clara del comportament del sistema. Així i tot, és convenient primer definir la màquina d'estats o autòmat que representa el comportament del dron. El diagrama d'estats definit pel dron es pot observar a la figura 3.2.

Com es pot observar al diagrama d'estat, la màquina d'estats s'inicia en l'estat inicial del qual només en surt quan detecta un esdeveniment (o missatge) de configuració. És llavors

quan passa a l'estat de configuració, que automàticament redirigeix l'estat a *ready*: en aquest estat el dron està a punt per començar a volar. A partir d'aquí, l'usuari pot ordenar al dron de començar a funcionar en dos possibles modes: el mode manual i el mode automàtic. En el mode manual, és l'usuari qui a través de l'estació terrestre comanda directament l'acció dels motors, en canvi en mode automàtic l'usuari especifica un punt de referència (és a dir, l'alçada desitjada) i el dron es basa en el seu sistema de control per a determinar a cada moment la potència necessària als motors per assolir la consigna.

Val a dir que pel que fa a l'estat `landingS`, durant el qual el dron duu a terme l'aterratge, s'hi pot arribar des d'una altra via a part dels esdeveniments de *stop*, i és que en el seu moment es va considerar la implementació d'un mecanisme d'aturada d'emergència, per la qual cosa quan es compleixen unes condicions determinades que es consideren alarmants, el dron aterra automàticament.

Els detalls relacionats amb la màquina d'estats, com les accions concretes que es realitzen a cada estat, es poden derivar de les funcions que en definitiva acaben implementant aquesta màquina d'estats. Així doncs, passem a explicar les diferents funcions que inclou el mòdul principal del nostre programa, en l'ordre en què es criden com a part de la rutina (s'ha cregut que d'aquesta manera explicant les funcions també queda il·lustrat el funcionament de la rutina que s'executa a cada volta del bucle):

1) `void setup()`

   Aquesta funció, que només s'executa un cop al començament del programa, crida les corresponents funcions d'inicialització dels diferents mòduls i configura alguns elements més, com ara el temporitzador que ha de generar la interrupció i l'estat inicial de l'autòmat implementat.

2) `void eventAnalyzer()`

   Aquesta funció és l'encarregada de consultar el port sèrie per a rebre els possibles missatges que s'hagin transmès des de l'estació terrestre. La manera com s'interpreten aquests missatges és la següent: es considera que el primer byte és el codi que indica quin tipus de missatge es tracta (els diferents tipus de missatges estan definits com a enumeració al codi del programa principal), i la resta de bytes s'interpreten com una estructura de dades del tipus `controllerData_t`.

   Aquests diferents tipus de missatges es consideren de cara a la màquina d'estat com a esdeveniments externs, que es preparen per a la seva interpretació més endavant com a part de la rutina del bucle principal.

   La manera com es llegeix la resta del paquet és la següent: Quan el nombre de caràcters llegits del port sèrie és igual a la mida d'aquest tipus d'estructura es considera correcte el missatge, i es passa a copiar el seu contingut a la variable que conté les dades d'estat del dron en si.

   Un problema d'aquest enfocament que a priori pot passar desapercebut és que si es comença a consultar el port sèrie quan encara no s'ha rebut un missatge sencer de l'estació terrestre hi ha el perill de què s'acabi descartant aquell missatge per no complir amb la mida especificada. Això, al seu torn, farà que el següent paquet també es descarti perquè no tindrà la forma esperada (amb el byte al principi que indiqui el tipus de missatge. Així i tot, es va comprovar que a la pràctica aquesta pèrdua puntual de missatges no té un impacte real en el funcionament global del sistema, i donat que el desenvolupament

d'un protocol de comunicació robust no quedava dins l'abast d'aquest projecte, es va donar considerar suficientment bona aquesta implementació.

3) `void emergencyAnalyzer()`

Aquesta funció realitza la tasca de detectar les condicions que han de dur el dron a realitzar un aterratge d'emergència, tal com es comentava més amunt. Finalment, el criteri que es va decidir utilitzar per a definir què es considera una situació d'emergència va ser el següent: si la velocitat dels motors ha estat igual a la màxima permesa durant un cert període de temps, que està parametritzat, s'imposa que l'estat passi a ser el d'aterratge (ignorant els possibles esdeveniments que s'hagin donat, ja que aquest estat només s'abandona quan s'ha arribat al final de la cursa del dron, sense possibilitat d'interrompre'l), que al diagrama d'estats apareix amb el nom extret del codi, `landingS`.

4) `void stateChangeAnalyzer()`

Aquesta funció analitza l'estat actual i calcula si es donen les condicions necessàries per a tenir lloc un canvi d'estat en funció d'aquest i els esdeveniments que s'hagin pogut rebre.

5) `void stateActionAnalyzer()`

Aquesta funció analitza l'estat actual del sistema i executa les instruccions necessàries en funció d'aquest. Algunes d'aquestes instruccions no estan reflectides al diagrama d'estats, però així i tot es considera que són prou rellevants per a comentar-les. Tenim, per tant, les següents accions a destacar donats els diferents estats possibles del dron:

a) `initialS`
   i) En aquest estat no es duu a terme cap acció

b) `setupS`
   i) Còpia de les dades rebudes pel programa principal per a configurar la instància de la classe `Controller`
   ii) Reescriptura de la distància de partida, que s'utilitza com a referència per a sortir de l'estat d'aterratge

c) `autoS`
   i) Actualització de l'estat del `Controller` especificant el mode automàtic i passant com a paràmetre el `setPoint` (referència d'alçada) rebut de l'estació

d) `manualS`
   i) Actualització de l'estat del `Controller` especificant el mode manual i passant com a paràmetre el `command` (consigna de velocitat dels motors) rebut de l'estació

e) `landingS`
   i) Lectura i comprovació de si la posició del dron és superior a la d'aterratge
   ii) Si és així, disminució progressiva de la velocitat dels motors (per a evitar caigudes amb excessiva velocitat)
   iii) Quan es deixa de complir, restabliment de l'estat `readyS`: el dron torna a estar a punt

f) `calibrationS`

i) Seqüència de cal·libració dels motors del dron, primer un cert període al nivell màxim i després al nivell mínim

L'estat de cal·libració es tracta d'una funcionalitat extra que no s'ha arribat a incorporar de manera que sigui accessible des del programa de l'estació terrestre, s'ha considerat que amb fer la cal·libració una vegada, si cal amb eines de *debug* que permetin crear els missatges necessaris per a iniciar la seqüència, ja és suficient.

6) `void readControllerData()`

Aquesta funció llegeix la informació de l'estat del controlador per a tenir una còpia actualitzada de les dades que hagin pogut canviar (sobretot com a resultat dels càlculs realitzats com a part del control, com ara l'estat de l'integrador o el derivador en el nostre cas, que incorpora un controlador PID).

7) `void sendData()`

Aquesta funció escriu al port sèrie la informació continguda a la variable d'estat amb tota la informació sobre el dron per tal de transmetre-la a l'estació terrestre.

I amb això donem per acabada la secció dedicada a descriure el programari injectat al dron. Tot i això, és evident que molts dels conceptes i decisions pel que fa al disseny del sistema que s'han explicat en aquest apartat tenen relació amb què s'explicarà a continuació, ja que l'estació terrestre s'ha desenvolupat tenint en compte el funcionament del dron per a permetre la comunicació i, al final, l'estreta relació entre els dos elements que han de fer d'aquesta plataforma (formada per la maqueta i l'estació terrestre) una eina útil i integral.

3.2 Estació terrestre

Com ja s'ha comentat, l'estació terrestre del nostre dron és el programa que dóna instruccions al dron mentre aquest està en funcionament. En el nostre projecte s'ha escollit utilitzar el llenguatge de programació d'alt nivell `Python` juntament amb la llibreria `TkInter` per a desenvolupar un programa amb interfície gràfica que permeti comandar el dron de manera eficaç i intuïtiva.

A l'hora de dissenyar el programa s'ha tingut sobretot present l'objectiu que perseguia aquest: oferir una eina complementària a la maqueta del dron que permeti agilitzar l'estudi dels efectes del control sobre la planta. Per tant, s'ha procurat que aquest programa oferís algunes funcionalitats que s'han considerat especialment desitjables: la modificació dels paràmetres del control en "calent", mentre el dron està en funcionament, la visualització numèrica i gràfica dels resultats mostrant les dades en temps real i l'opció de prendre mostres gravant els resultats per al seu tractament posterior.

A continuació s'expliquen els diferents mòduls en què s'ha organitzat el codi per fer-lo més entenedor i clar, i la manera com aquests diferents mòduls es relacionen. Com a punt de partida, a la figura 3.3 tenim el diagrama de blocs del programa.

Les relacions entre els diferents mòduls que poden no ser òbvies a primera vista s'expliquen més en detall als següents apartats, que descriuen les funcions essencials de les classes contingudes a cada un d'aquests.

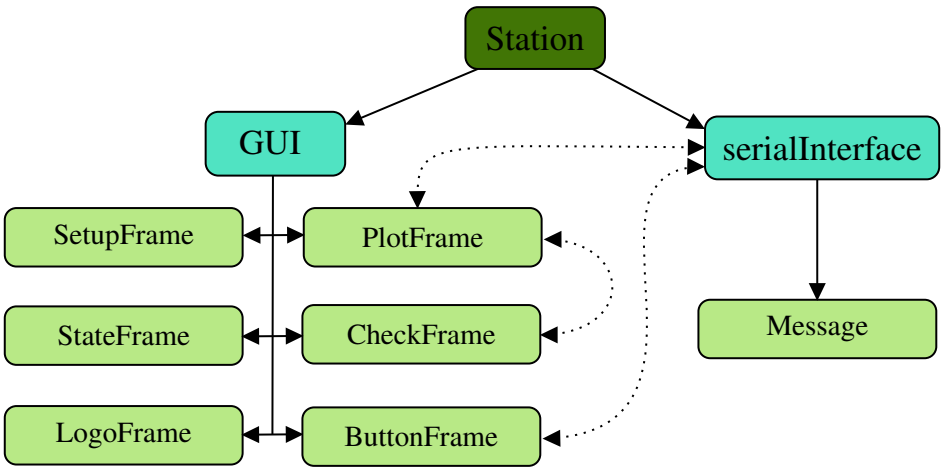

Figura 3.3: Diagrama de blocs del programa de l'estació terrestre.

3.2.1 Interfície gràfica

L'interfície gràfica de l'estació terrestre s'ha desenvolupat utilitzat la llibreria pròpia de `Python` *TkInter* ([**TkInter**]. El motiu de l'elecció d'aquest *framework* o marc de treball va ser que es tracta d'una eina explorada durant el grau i que, si bé potser no és la més moderna i potent del mercat, inclou tots els ginys que poden ser d'utilitat en el nostre programa (no en va es considera la llibreria *de facto* per a les interfícies gràfiques desenvolupades amb `Python`).

Un aspecte essencial d'aquesta llibreria és que requereix que el codi tingui una estructura particular, que bàsicament consisteix en la creació de totes les estructures que formaran part de la GUI (de l'anglès *Graphical User Interface*) de manera prèvia i finalment la crida a una funció que actua com a moderadora gestionant tots els esdeveniments i interaccions entre l'usuari i la interfície gràfica. Aquesta funció és l'anomenada `mainloop()`.

Aquests elements estructurals que s'han de definir com a part de la configuració inicial de la interfície poden ser de diversos tipus i organitzar-se en l'espai d'acord amb diferents estratègies. El bloc principal que ha actuat com a separador de les diferents parts del nostre programa és el *frame*, d'acord amb la seva nomenclatura en anglès, que en català traduïm com a "marc". Per simplicitat i modularitat del codi, s'han organitzat els diferents mòduls del programa que fan referència a la interfície gràfica de manera que cadascun contingui una classe derivada de la classe que `TkInter` ofereix per als marcs, de manera que de fet cada mòdul acaba representant una part concreta del nostre programa. Així doncs, els diferents elements del nostre programa es poden explicar detallant els continguts de cada un d'aquests mòduls, cosa que es fa a continuació.

Mòdul `SetupFrame`

Aquest marc principalment defineix els camps d'introducció de text per part de l'usuari que després s'utilitzen per a configurar el dron amb els valors introduïts.

El marc de configuració també inclou un lliscador que permet ajustar o bé el valor de la velocitat dels motors (quan el dron està en mode manual) o bé la referència d'alçada desitjada (quan està en mode automàtic). Aquest lliscador està desactivat mentre el dron no està a punt per funcionar, però en canvi sí que es permet operar-lo quan el dron està en l'estat `readyS`;

d'aquesta manera es pot aconseguir que la referència d'alçada, que en el nostre llaç de control equival a l'entrada del sistema, pugui tenir la forma d'un graó (que val zero fins al moment inicial en què passa a tenir un valor determinat), una característica que pot ser interessant per tal d'observar la resposta del dron a aquest tipus d'impulsos. Això sí, aquesta funcionalitat només s'aplica quan s'activa el mode automàtic, ja que per seguretat es va considerar que en mode manual el dron sempre ha de començar al valor mínim de velocitat dels motors.

Així doncs, aquest mòdul ofereix la següent classe i funcions (de fet, per ser precisos, mètodes d'aquestes classes):

1) `class SetupFrame(tk.Frame)`
   a) `def __init__(self, parent, *args, **kwargs)`
   b) `def resetScale(self)`
   c) `def disableScale(self)`
   d) `def enableScale(self)`
   e) `def setScaleRange(self, from_, to)`
   f) `def getParameters(self)`

Com es pot observar, la classe que dóna nom al mòdul incorpora diverses funcions que altres mòduls al seu torn utilitzen per a manipular el lliscador (activant-lo, desactivant-lo o modificant-ne el rang) i per obtenir els paràmetres de configuració del dron introduïts per l'usuari. La manera com s'envia el valor desitjat per a la referència d'alçada o la velocitat dels motors (en funció del mode d'operació) indicat amb el lliscador és mitjançant una funció periòdica que va creant paquets amb aquesta informació per enviar-los al dron, i que es va executant repetidament mentre el dron (i per tant l'estació, que en coneix l'estat) està en marxa.

Pot ser interessant comentar que per a reduir el temps fins que es posa en marxa el dron un cop s'inicia el programa, es va implementar una funcionalitat afegida consistent en uns valors predeterminats que es poden utilitzar com a punt de partida per a fer la configuració inicial del dron.

L'aspecte final d'aquest marc de la interfície gràfica es pot observar a la figura 3.4.

Mòdul `PlotFrame`

Aquest mòdul implementa les classes involucrades en la visualització en si de les dades obtingudes per l'estació. Per a la creació de les diferents gràfiques mostrades al sistema es va optar per utilitzar la llibreria de `Python` `matplotlib` ([**matplotlib**]), una popular llibreria de gràfiques i animacions. Aquesta llibreria s'ha hagut d'integrar com a part de l'estructura de `TkInter`, cosa que ha estat possible gràcies al suport que existeix per aquesta llibreria des del marc de treball de la interfície gràfica.

A continuació tenim un índex de les diferents classes definides en aquest mòdul i els mètodes que incorporen:

1) `class CustomNavigationToolbar(NavigationToolbar2Tk)`
   a) `def __init__(self, canvas_, parent_)`
   b) `def home(self)`

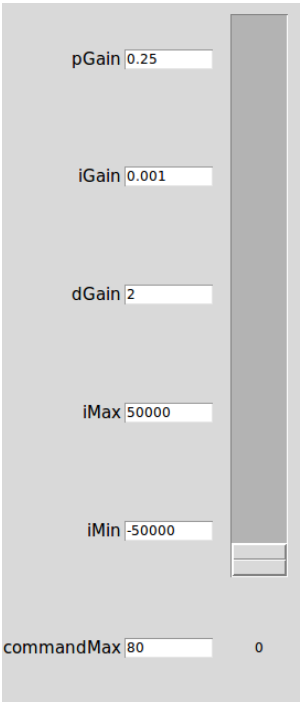

Figura 3.4: Marc de configuració, definit al mòdul `SetupFrame`.

c) `def pan(self)`

d) `def zoom(self)`

e) `def save_figure(self)`

2) `class PlotFrame(tk.Frame)`

a) `def __init__(self, parent, *args, **kwargs)`

i) `def plotFunction(frame)`

b) `def enableAutoscale(self)`

c) `def disableAutoscale(self)`

d) `def stop(self)`

e) `def resume(self)`

f) `def closePlot(self)`

Com es pot observar, el mòdul inclou una subclasse que hereta de `NavigationToolbar2Tk`, la classe de la llibreria `matplotlib` que s'utilitza per a representar un objecte de barra de navegació dins el gràfic com a element propi l'entorn de `TkInter`. Els diferents mètodes d'aquesta classe s'han reescrit per a personalitzar-ne l'acció permetent, per exemple, desplaçar-se o ampliar o reduir zones del gràfic mentre aquest es continua actualitzat, cridant les corresponents funcions de la següent classe.

La classe que representa el marc de les gràfiques en si hereta, com la resta de marcs, de la classe de `tk.Frame`. Aquest marc inclou la figura en què es dibuixen les diferents gràfiques, i

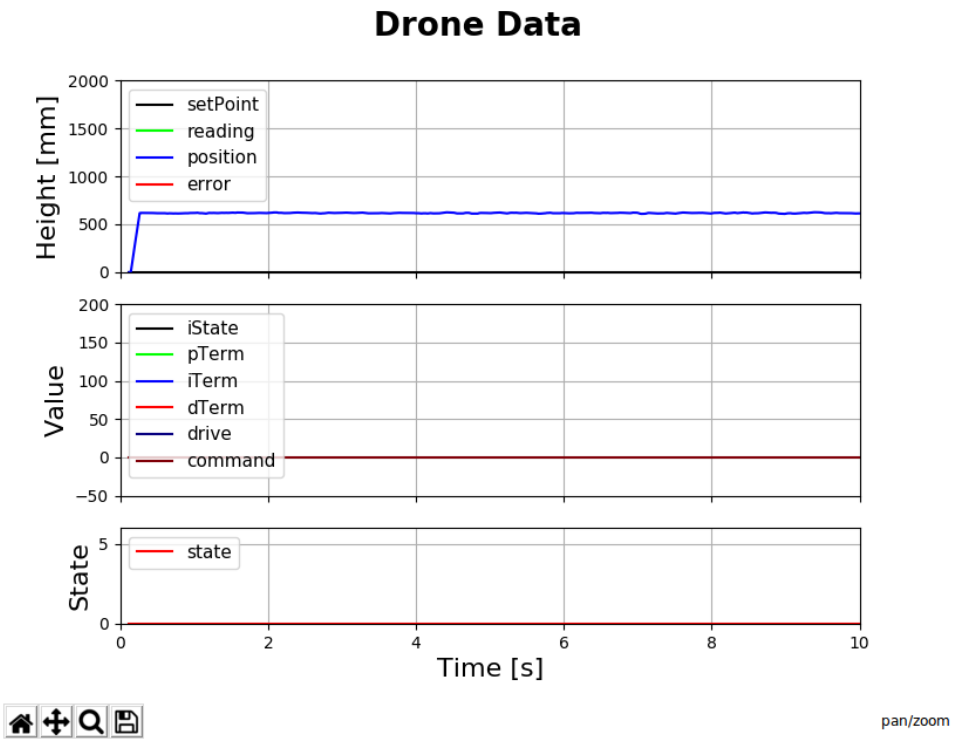

Figura 3.5: Marc de visualització, definit al mòdul `PlotFrame`.

ofereix les funcions cridades per la barra de navegació per a activar o desactivar l'actualització i l'escalat automàtic dels gràfics.

Una de les relacions entre mòduls que s'indica al diagrama de mòduls és la que té lloc entre aquest mòdul i el mòdul `CheckFrame`. Aquest últim mòdul conté els ginys que permeten a l'usuari seleccionar quines variables vol visualitzar (organitzades en diferents gràfiques d'acord amb el tipus i magnitud d'aquestes variables). La manera com s'aconsegueix això és seleccionant les llistes de dades a dibuixar en funció dels valors de les diferents caselles del marc `CheckFrame`, que es llegeixen d'aquest objecte accedint-hi a través de la unió dels dos mòduls amb el programa principal.

L'aparença final del marc de visualització es pot observar a la figura 3.5.

Mòdul `CheckFrame`

Aquest mòdul solament implementa la classe corresponent al marc de selecció que inclou les caselles que permeten escollir quines variables extretes de les dades del dron es volen mostrar al marc de visualització.

Aquest mòdul, doncs, només incorpora una classe:

1) `class CheckFrame(tk.Frame)`
    a) `def __init__(self, parent, *args, **kwargs)`

La manera com aquesta classe interactua amb el marc de visualització s'explica en l'apartat anterior, si bé bàsicament consisteix en el fet que aquesta última classe accedeix als atributs

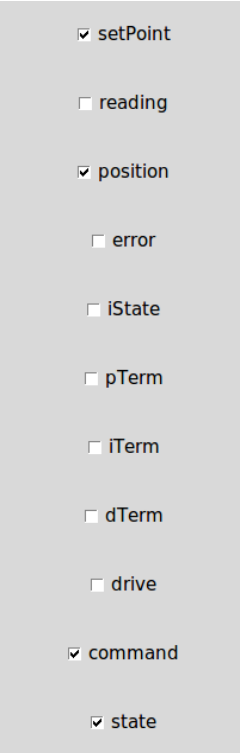

Figura 3.6: Marc de selecció, definit al mòdul `CheckFrame`.

de les caselles de selecció pel camí que uneix els dos mòduls (a través del programa principal, el mòdul `Station`. És interessant notar com, de manera semblant com passava amb el marc de configuració, s'han definit un seguit de caselles que s'inicialitzen com a marcades de manera predeterminada per a comoditat de l'usuari (s'ha considerat que les variables marcades són les que a priori poden resultar més interessants a primer cop d'ull).

L'aparença final del marc de selecció es pot observar a la figura 3.8.

Mòdul `ButtonFrame`

Aquest mòdul solament implementa la classe corresponent al marc dels botons utilitzats per a controlar les accions principals del dron: configuració, encesa i aturada.

Aquest mòdul, doncs, només incorpora una classe:

1) `class ButtonFrame(tk.Frame)`
    a) `def __init__(self, parent, *args, **kwargs)`

Els diferents botons s'activen o es desactiven depenent del moment en funció de l'estat del dron rebut com a part de les dades provinents de la planta (per exemple, el botó d'aturada només està activat quan el dron està encès, ja que d'altra manera no tindria sentit). Cada botó té associada una funció del mòdul principal que al seu torn crida el mòdul encarregat de gestionar la comunicació sèrie per a enviar les instruccions convenients al dron. Aquesta relació indirecta és la representada al diagrama entre aquests dos mòduls.

L'aparença final del marc dels botons es pot observar a la figura 3.9.

Figura 3.7: Marc dels botons, definit al mòdul `ButtonFrame`.

Mòdul `CheckFrame`

Aquest mòdul solament implementa la classe corresponent al marc de selecció que inclou les caselles que permeten escollir quines variables extretes de les dades del dron es volen mostrar al marc de visualització.

Aquest mòdul, doncs, només incorpora una classe:

1) `class CheckFrame(tk.Frame)`
    a) `def __init__(self, parent, *args, **kwargs)`

La manera com aquesta classe interactua amb el marc de visualització s'explica en l'apartat anterior, si bé bàsicament consisteix en el fet que aquesta última classe accedeix als atributs de les caselles de selecció pel camí que uneix els dos mòduls (a través del programa principal, el mòdul `Station`. És interessant notar com, de manera semblant com passava amb el marc de configuració, s'han definit un seguit de caselles que s'inicialitzen com a marcades de manera predeterminada per a comoditat de l'usuari (s'ha considerat que les variables marcades són les que a priori poden resultar més interessants a primer cop d'ull).

L'aparença final del marc de selecció es pot observar a la figura 3.8.

Mòdul `ButtonFrame`

Aquest mòdul solament implementa la classe corresponent al marc dels botons utilitzats per a controlar les accions principals del dron: configuració, encesa i aturada.

Aquest mòdul, doncs, només incorpora una classe:

1) `class ButtonFrame(tk.Frame)`
    a) `def __init__(self, parent, *args, **kwargs)`

Els diferents botons s'activen o es desactiven depenent del moment en funció de l'estat del dron rebut com a part de les dades provinents de la planta (per exemple, el botó d'aturada només està activat quan el dron està encès, ja que d'altra manera no tindria sentit). Cada botó té associada una funció del mòdul principal que al seu torn crida el mòdul encarregat

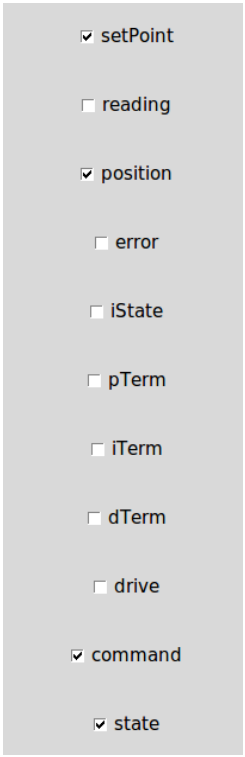

Figura 3.8: Marc de selecció, definit al mòdul `CheckFrame`.

de gestionar la comunicació sèrie per a enviar les instruccions convenients al dron. Aquesta relació indirecta és la representada al diagrama entre aquests dos mòduls.

L'aparença final del marc dels botons es pot observar a la figura 3.9.

3.2.2 Interfície sèrie

El canal de comunicació entre l'estació terrestre i el dron en si es tracta d'una interfície o port sèrie, com ja s'ha comentat en el referent al codi injectat al dron. Es va escollir aquest tipus de canal ja que ofereix moltes comoditats de cara a l'usuari, permetent també que l'eina docent que pretén ser aquest projecte sigui més accessible. Aquestes comoditats es redueixen, bàsicament, al fet que tant la mateixa placa *Arduino* com la immensa majoria (si no tots) els ordinadors disposen d'un port USB al qual es pot connectar un cable d'aquest mateix tipus que funcioni com a canal de comunicació.

Si bé com tots els canals de comunicació un port sèrie també té un cabal limitat pel que fa a la quantitat de dades per unitat de temps que es poden transmetre a través d'aquest, ben aviat es va detectar que això no seria una limitació en el nostre projecte.

Tal com s'ha explicat a l'apartat corresponent al codi del dron que gestiona la comunicació sèrie, el bloc que s'ha definit com a unitat de transmissió en el nostre protocol rudimentari és un paquet de 60 bytes que conté totes les dades referents a l'estat del dron, entre els quals la posició, els paràmetres del control o la marca de temps. A aquest bloc s'hi afegeix un byte extra quan la direcció de la comunicació és de l'estació al dron, que correspon al codi de la instrucció o esdeveniment que s'està enviant al dron.

Figura 3.9: Marc dels botons, definit al mòdul `ButtonFrame`.

Els missatges des del dron fins a l'estació, que com s'ha dit són paquets de 60 bytes, s'envien a la freqüència marcada per la de mostreig del dron, això són 50,00 Hz. Per altra banda, els missatges de l'estació cap al dron s'envien amb màxima freqüència quan aquest està en funcionament (en aquest context l'estació constantment envia missatges amb els nous valors introduïts per l'usuari a través del lliscador); aquesta freqüència (que es pot modificar canviant un paràmetre del codi) al llarg del desenvolupament ha estat també de 50,00 Hz.

Amb aquesta informació i cert coneixement extra sobre el protocol sèrie, que marca que cada byte transmès va acompanyat de tres bits extres (un de *start* i dos de *stop*), podem calcular el cabal màxim utilitzat al canal de comunicació de la següent manera:

$$cabal_{max} = (60 + 61) * 50 * 11 = 66550$$

Tenim que el cabal màxim a la connexió sèrie és de 66 550,00 bit/s. Tenint en compte que la placa *Arduino* permet velocitats de fins a 115 200,00 bit/s per al port sèrie, tenim que només estem utilitzant aproximadament el 60% de la capacitat del canal en els moments de màxima demanda. Així doncs, efectivament podem confirmar que la capacitat de la connexió no ha estat una limitació en el nostre projecte, per la qual cosa s'ha optat per tenir un protocol de comunicació més rudimentari, més accessible que no pas òptim o eficient, però que sigui funcional i útil de cara a l'usuari (pensant en estudiants que puguin voler treballar amb aquesta maqueta i fins i tot modificar-ne alguns aspectes).

A continuació, doncs, s'expliquen els detalls d'implementació rellevants pel que fa al mòdul de gestió de la interfície sèrie com a part del programa de l'estació terrestre.

Mòdul `Message`

Aquest mòdul implementa les classes corresponents als tipus de missatge del protocol de comunicació desenvolupat per al projecte: per una banda tenim els missatges de l'estació al dron (representats per la classe `ControllerMessage` i per l'altra els del dron a l'estació (representats per la classe `ControllerData`), ambdós missatges heredant característiques del bloc de dades en general (representat, al seu torn, per la superclasse `Message`). L'índex de les classes i mètodes que implementa el mòdul, doncs, és el següent:

1) `class Message()`
    a) `def __init__(self, name, prefix, fields)`
    b) `def info(self)`
    c) `def getField(self, field)`
    d) `def setField(self, field, value)`
    e) `def __str__(self)`
2) `class ControllerData(Message)`
    a) `def __init__(self)`
    b) `def unpack(self, bytes)`
3) `class ControllerMessage(Message)`
    a) `def __init__(self)`
    b) `def pack(self)`

Com es pot observar, les dues classes per als missatges del protocol bàsicament ofereixen mètodes per a "empaquetar" les dades que es volen enviar des de l'estació en forma de bytes i per a "desempaquetar" els bytes rebuts des del dron en forma de dades.

Mòdul `SerialInterface`

Aquest mòdul implementa la classe `SerialInterface`, que ofereix els mètodes necessaris per a comunicar-se amb el dron a través del port sèrie i per a extreure la informació concreta continguda als paquets rebuts.

1) `class SerialInterface()`
    a) `def __init__(self, station, port, baud)`
    b) `def start(self)`
    c) `def backgroundThread(self)`
    d) `def stop(self)`
    e) `def getState(self)`
    f) `def send(self)`
    g) `def close(self)`

La manera com s'aconsegueix anar rebent els missatges del port sèrie a mesura que el dron els envia al marge del bucle principal de la interfície gràfica (que d'altra manera "segrestaria" l'execució del programa) és delegant les funcions de lectura del port a un altre fil del programa o *thread* en anglès, que constantment i fins que no s'atura va llegint del port sèrie i interpretant les dades en funció de l'estructura coneguda dels paquets. Amb aquesta informació extreta dels paquets entrants s'afegeixen les dades a les llistes des d'on la classe `PlotFrame` les extreu per a visualitzar-les per pantalla.

3.2.3 Mòdul `Station`

Aquest mòdul actua com a programa principal, i per tant integra la resta d'elements per a coordinar-ne el funcionament.

El mòdul implementa una única classe que representa el conjunt de l'estació terrestre, i un seguit de mètodes que orquestren les diferents operacions realitzades pel programa:

1) `class Station()`
    a) `def __init__(self, port, baud)`
    b) `def start(self)`
    c) `def setupCommand(self)`
    d) `def stopCommand(self)`
    e) `def autoScaleCommand(self)`
    f) `def manualScaleCommand(self)`
    g) `def autoCommand(self)`
    h) `def manualCommand(self)`
    i) `def updateState(self)`
    j) `def close(self)`

Els diferents mètodes de la classe es corresponen a operacions molt concretes i clares pel que fa a la seva intenció.

Per una banda, tenim les funcions que inicien i finalitzen l'execució del programa, cridant a les funcions semblants dels diferents mòduls i assegurant-se que s'han gestionat tots els esdeveniments abans de finalment tancar el programa.

Per l'altra banda, tenim les funcions que es corresponen a les crides que realitzen els botons de la classe `ButtonFrame` quan aquest són activats, i per tant tenim una funció per cada acció del dron (configuració, encesa en mode manual, encesa en mode automàtic i apagat). Aquestes funcions duen a terme totes les accions necessàries als diferents mòduls del programa per mantenir la coherència en l'estat.

Finalment, tenim la funció d'actualització de l'estat, anàloga a la rutina principal del programa del dron, que periòdicament s'executa (de manera temporitzada pel gestor d'esdeveniments de l'entorn gràfic) per tal de llegir les noves dades rebudes pel port sèrie i actuar en conseqüència.

4 Control del sistema

En aquest capítol es descriuen els aspectes del projecte relacionats amb el control en si, i amb les simulacions realitzades per tal d'intentar modelar i en definitiva, comprendre, el comportament del sistema desenvolupat.

És important entendre que d'alguna manera aquest apartat pretén il·lustrar la utilitat de la plataforma docent desenvolupada com a eina per a l'estudi dels conceptes relacionats amb el control. Així, es busca realitzar un procés d'exploració de la planta i del control que s'hi podria implementar tal com s'anima a fer als usuaris d'aquesta eina. És per això que no s'aprofundeix excessivament en la teoria del control sinó que es parteix d'una base relativament simple per extreure aprenentatges a través de l'experimentació.

Dit això, passem a comentar breument l'estructura d'aquest capítol.

En primer lloc es fa una breu introducció als conceptes generals al voltant de les estructures de control, posant especial èmfasi en un tipus de controlador molt utilitzat en drons, el controlador PID. En aquesta introducció també es parla del comportament desitjat per al dron de la maqueta, i de quina manera s'utilitza un controlador PID per intentar donar aquest comportament a la planta.

Seguidament, es passarà a descriure els elements que s'han tingut en compte a l'hora de desenvolupar un model del sistema que ens permeti estudiar el seu comportament per mitjà de la simulació. Aquí parlarem del modelatge de la planta i l'impacte de les diferents variables referents al control sobre el comportament de la planta, per acabar avaluant el funcionament del dron físic de la maqueta comparant-lo amb les simulacions fetes.

4.1 Estructures de control

La teoria de control és un camp de les matemàtiques aplicades a l'enginyeria que estudia el comportament dels sistemes per desenvolupar estructures que permetin controlar-los, és a dir, assolir els paràmetres desitjats amb la màxima rapidesa, suavitat i estabilitat possible (fent referència a [**control**]). D'alguna manera, els reptes que es plantegen (que al seu torn s'exploren en diferents branques d'aquest camp de coneixement), són el modelatge dels sistemes, la capacitat d'observabilitat d'aquests i finalment, el control.

Aquests objectius del control s'apliquen al nostre dron de la manera següent: per començar, el dron és un sistema mecànic amb unes certes limitacions i un comportament dinàmic que es pot analitzar per conèixer els paràmetres que caracteritzen el seu moviment. Per tal d'acabar tenint un control precís i eficaç, és important que les estructures de control puguin conèixer l'estat de la planta en un moment determinat, per la qual cosa és necessari que el dron incorpori algun tipus de sensors que mesurin aquesta informació. Finalment, es pretén aconseguir poder governar el dron i que aquest compleixi de manera satisfactòria amb les consignes donades: volem poder indicar una alçada desitjada i que el dron s'adapti per a estabilitzar la seva posició a aquesta mateixa alçada.

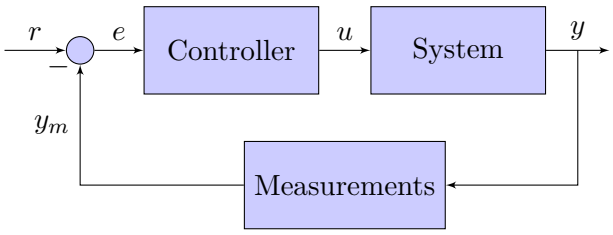

Figura 4.1: Diagrama bàsic d'un sistema de control de llaç tancat.

L'estructura típica que permet resumir la relació entre aquests elements involucrats en el control d'un sistema és l'anomenat "control de llaç tancat", un tipus de sistema caracteritzat per una planta i un controlador que s'alimenta de les dades obtingudes sobre la planta per ajustar-ne el comportament. El diagrama d'aquesta típica estructura de control es pot observar a la figura 4.1.

En el nostre cas s'ha volgut partir d'un dels controladors més utilitzats, i potser del millor exemple dels controladors de llaç tancat, per aprendre millor el seu funcionament i observar-lo a la pràctica. Aquest controlador, que s'explora més en detall a continuació, es tracta del controlador PID , que deu el seu nom als elements de proporcionalitat, integració i derivació que incorpora.

4.1.1 Controlador PID

El controlador PID es tracta d'un exemple típic de control amb llaç tancat. El seu funcionament bàsic, que no s'allunya del comportament il·lustrat al diagrama d'un controlador de llaç tancat genèric que s'ha vist abans, és el següent: l'entrada del control $e(n)$ es calcula com la diferència entre la referència desitjada $r(n)$ (que és l'entrada del sistema global) i el valor actual de la sortida de la planta $y_m(n)$ (escalada d'acord amb el sistema de mesura corresponent). Aquesta diferència s'utilitza com a entrada dels tres components que li donen el nom al controlador: per una banda hi ha el component proporcional, que es calcula directament a partir de l'error actual, després tenim el component integrador que es calcula de manera proporcional a l'error acumulat i finalment el component derivador que es calcula de manera proporcional a la diferència entre dues mostres consecutives.

La manera com es calcula la sortida del control considerant una implementació digital d'aquest, es pot expressar de la manera següent:

$$u(n) = k_p * e(n) + k_e * \sum_{i=0}^{n} e_i + k_d * (e(n) - e(n-1))$$

on

$$e(n) = r(n) - y_m(n)$$

Aquesta sortida del control, $u(n)$, sovint anomenada senyal de *drive* del sistema, és l'entrada de la planta, i per tant és la responsable de causar un canvi en l'estat del sistema que queda reflectit a la sortida de la planta $y(n)$ d'acord amb una determinada relació, que a vegades s'anomena la funció de transferència de la planta. Aquesta funció de transferència, que representa el comportament de la planta donada una determinada entrada, pot dependre d'un gran nombre de variables, i la seva determinació quantitativa sovint pot resultar ser massa

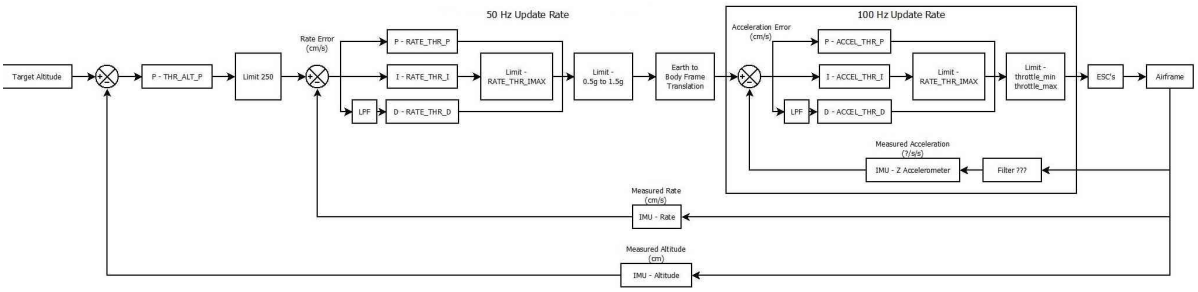

Figura 4.2: Diagrama del controlador per a drons del projecte *Ardupilot*. Font: *Ardupilot*.

imprecisa, de manera que un coneixement qualitatiu d'aquesta ja pot ser útil per a estudiar el seu comportament.

L'estructura que acabem de descriure permet, dins d'uns certs límits, que donats uns paràmetres dels diferents components del controlador PID , parlant de k_p, k_i i k_d, sigui possible tenir una sortida relativament estable i ràpida que segueixi la referència introduïda al sistema. Això, però, com es pot imaginar sovint no resulta tan senzill, i és necessari un afinament precís dels paràmetres, que tot i això pot no sempre ser suficient.

L'eficàcia del sistema de control depèn en gran part del comportament de la planta, i per això un mateix sistema de control pot ser útil per a la seva aplicació en un tipus de sistemes però no ser-ho en un altre. Concretament, el controlador PID s'utilitza sovint en sistemes mecànics que requereixen un reajustament freqüent, característiques que coincideixen amb les necessitats de la nostra maqueta.

Basant-nos en l'ús recorrent del controlador PID com a estructura típica de control per a sistemes mecànics, i buscant un punt de partida per a desenvolupar un control que permeti governar el nostre dron de manera més o menys efectiva, doncs, es va optar per implementar aquest tipus de controlador al codi del dron, si bé queda la porta oberta a que en un futur algú altre modifiqui i millori aquesta implementació.

És interessant senyalar que algunes implementacions de sistemes de control per a drons sovint utilitzen estructures de control molt més complexes que un simple controlador PID , si bé també disposen de més sensors i de fet, també d'actuadors (un dron real té més motors que no pas el de la nostra maqueta). Així i tot, val a dir que sovint s'utilitzen estructures reminiscents o directament derivades de la d'un controlador PID , ja que és reconegut per la seva flexibilitat i capacitat d'ajustar-se a la planta fins i tot quan aquesta no es coneix del tot (cosa que sovint passa a la vida real).

Com a referència, a la figura 4.2 extreta de la documentació oficial del projecte *Ardupilot* ([**ardupilot**]) per a drons, es pot observar el diagrama del control que implementa aquest codi per a estabilitzar el dron a l'aire. Es pot observar com en realitat consisteix de tres controladors d'alguna manera "encadenats", que a grans trets consisteixen en un controlador P i dos PID , que controlen la sortida desitjada per a cada etapa (en aquest cas existeix *feeback* tant per la posició, com la velocitat i l'acceleració del dron).

A grans trets, i de manera intuïtiva, es pot fer una interpretació d'alt nivell pel que fa a les funcions de cada un dels components del controlador PID de la següent manera:

1) *P:* el component proporcional aporta un valor d'entrada a la planta que creix de manera proporcional a l'error, tot i això no és capaç d'adaptar-se en funció de l'evolució d'aquest

error i per tant mai arriba a eliminar-lo del tot —"com més lluny sóc d'on vull arribar, més li demano a la planta"

2) *I:* el component integratiu aporta una entrada a la planta que correspon a la integral de l'error, de manera que creix a mesura que hi ha error i fa que a la llarga s'assoleixi l'objectiu—"si fa estona que l'error no disminueix, li demano més a la planta"

3) *D:* el component derivatiu aporta una entrada a la planta que depèn de la variació de la sortida, per tant evita les variacions brusques —"si la sortida està canviant massa de pressa, m'hi oposo"

Aquesta visió simplificada no deixa de ser útil per analitzar el comportament d'un controlador d'aquest tipus, i és interessant tenir aquest punt de vista més intuïtiu a l'hora de valorar l'impacte d'incrementar o disminuir les constants que regeixen l'acció de cada un d'aquests components. En definitiva, aquesta visió ha estat de gran ajuda a l'hora d'interpretar resultats i buscar solucions per als problemes que han anat sorgint durant la implementació del control, que es descriuen més endavant.

4.2 Simulació

4.2.1 Modelatge de la planta

Tal com s'ha comentat, una part essencial a l'hora de dissenyar sistemes de control és el modelatge de la planta, ja que conèixer informació sobre com aquesta canvia el seu estat donades diferents entrades ens pot ajudar a dissenyar una estructura de control que sigui òptima. En el cas d'un controlador PID això ens pot orientar pel que fa al valor que han de tenir les constants, donant-nos un punt de partida per a l'ajustament d'aquests valors.

En el nostre cas, però, el coneixement sobre les característiques de la planta s'ha buscat per tal de ser capaços de tenir un model el més ajustat possible a la realitat per tal d'utilitzar-lo per a dur a terme simulacions sobre la resposta de la planta donades diferents entrades i estructures de control.

Amb aquest objectiu, es van realitzar diverses mesures de vàries magnituds físiques pròpies de la planta per mirar d'incorporar-les a les equacions ideals que descriurien el seu comportament. Fent referència a aquestes equacions, a continuació es detalla com s'han desenvolupat aquestes fins a tenir-les en la forma de la implementació en el llenguatge de programació orientat als càlculs matemàtics *Octave* ([**octave**]).

A l'hora de fer un model matemàtic del comportament de la planta cal tenir en compte que el disseny de la maqueta fixa que l'únic grau de llibertat que té el dron és el del moviment al llarg de l'eix vertical. Aquest moviment té una acceleració variable que depèn de la velocitat dels motors, que al seu torn és una funció del rendiment de cicle del senyal d'entrada dels motors. Per tant, per poder caracteritzar correctament la planta primer necessitem conèixer aquesta relació.

Per a mesurar aquesta relació es van desenvolupar dos programes simples, un programa escrit en C++ que es troba a l'arxiu `speedController.ino` que s'injecta al dron i que senzillament espera ordres per a enviar-les als controladors dels motors, i un programa escrit en el llenguatge de programació *Octave* que es troba a l'arxiu `frequencySampler.m` que, amb l'ajuda d'un micròfon, enregistra senyals auditius donades diferents velocitats de rotació dels motors per

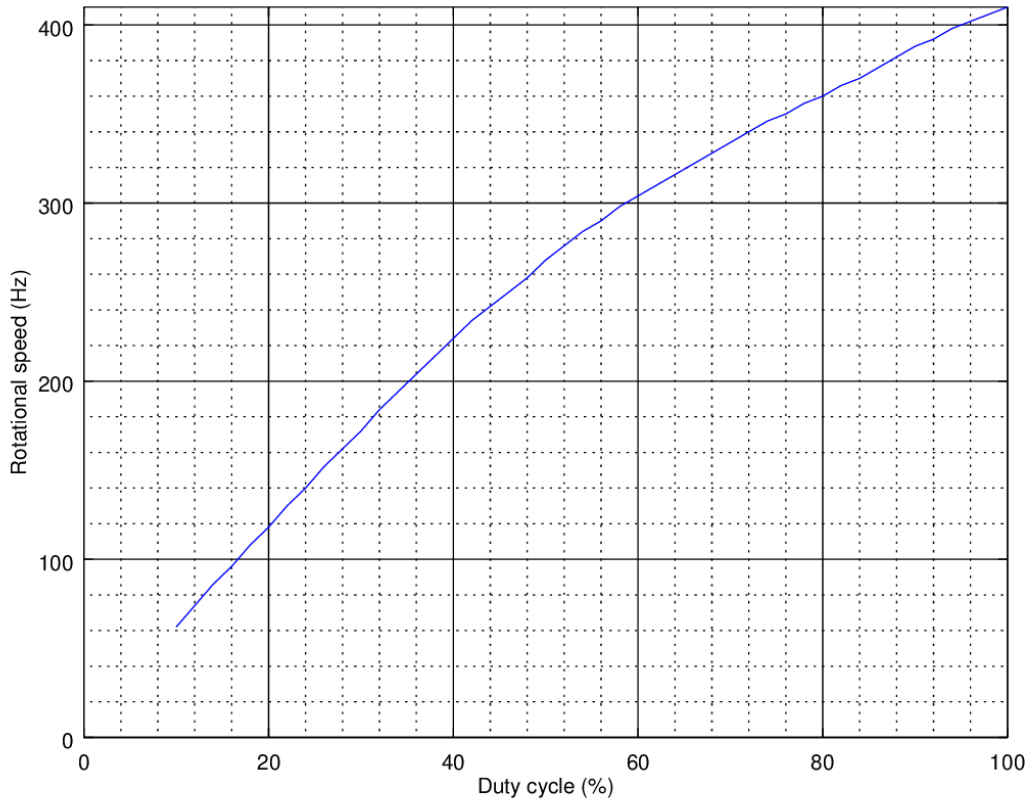

Figura 4.3: Gràfic que relaciona el rendiment de cicle del senyal a l'entrada del controlador amb la velocitat de rotació resultant al motor.

a detectar-ne les freqüències de nivell més alt. Aquestes freqüències corresponen al primer harmònic de la velocitat de rotació real (ja que donada la simetria de les hèlices pràcticament no hi ha diferència entre les dues meitats de la volta sencera del motor i per tant el component freqüencial principal és de fet el causat per aquestes mitges voltes), i ens permeten crear gràfiques que mostrin la relació entre el rendiment de cicle a l'entrada de la planta i la velocitat de rotació dels motors com a resposta. Els resultats d'aquestes mesures, realitzades amb els motors, els controladors i les hèlices descrits als apartats anteriors, es poden observar a la figura 4.3.

Com es pot observar a la gràfica, la relació no és exactament lineal, especialment a velocitats més altes del motor, quan sembla que la tendència és més pròxima a la gràfica d'una arrel quadrada. Això ens podrà ser útil per a contrastar els resultats obtinguts a continuació.

El següent pas, consistent en relacionar la força als motors en funció de la velocitat de rotació, va consistir en mesures manuals fetes amb dinamòmetres de diferents escales fixats al perfil d'alumini col·locat en posició horitzontal per determinar la força exercida pels motors en funció de la velocitat (que ara coneixem per cada valor de rendiment de cicle). Per a obtenir els valors reals de força també es van mesurar les forces de fregament i el pes de diferents elements del dron (inclosos els cables) per tal de contrastar els valors.

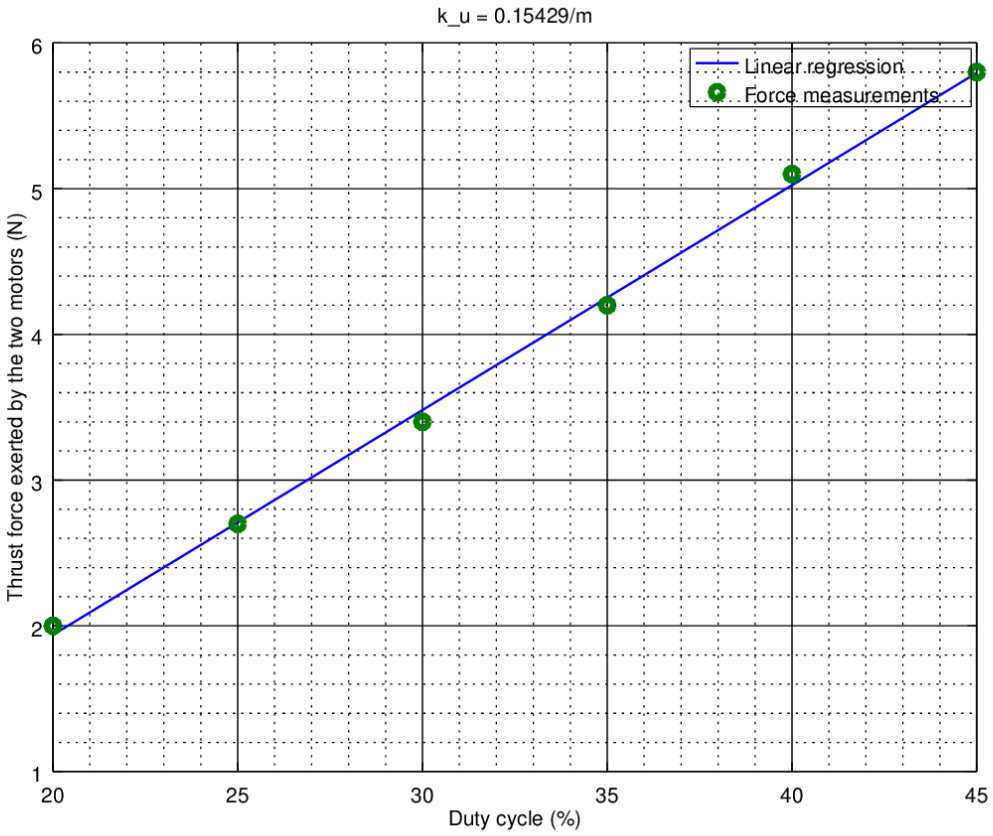

Figura 4.4: Gràfic que relaciona el rendiment de cicle del senyal a l'entrada del motor amb la força d'empenyiment resultant.

Després de prendre aquestes mesures, es van obtenir els resultats de la figura 4.4. Notar que només es tenen valors de les forces a velocitats de rotació relativament baixes, ja que a partir d'aquestes les eines de mesura no tenien prou rang per a extreure mesures vàlides.

Aquest cop la relació mostra un comportament més lineal. Tot i que diferents factors han pogut ser causants d'error en les mesures, i que possiblement les no linealitats introduïdes pels controladors dels motors (que podrien incorporar els seus propis sistemes de control i donar lloc a canvis de velocitat en funció de variacions a l'alimentació, la temperatura dels *drivers* o els motors, etc.) o el fet de no haver pogut prendre mesures en tot el rang de velocitats que pot tenir el motor, els resultats obtinguts poden tenir un interès qualitatiu, si bé no tan quantitatiu. Malgrat això, i donat que possiblement d'altres factors dificulten el modelatge de la planta, s'ha donat per bona aquesta relació de proporcionalitat entre la força generada pels motors i el rendiment de cicle a l'entrada d'aquests.

Així doncs ara tenim que podem definir l'acceleració total del sistema de la següent manera:

$$a(t) = k_u * u(t) - g(t)$$

On k_u és la constant de proporcionalitat obtinguda a la figura 4.4 que relaciona la força exercida pels motors amb el senyal d'entrada u, que pot prendre valors del 0 al 100 (equivalents

al rang de rendiment de cicle que s'interpreta als controladors dels motors), i on g és l'acceleració de la gravetat (que s'ha considerat en sentit contrari a la força exercida pels motors, que se li ha donat sentit positiu en l'eix vertical).

Coneixent la relació entre la posició del dron i la seva acceleració, podem escriure:

$$y(t) = \iint a(t)\ dt$$

Si integrem l'expressió i l'estudiem en el domini discret, podem obtenir unes expressions que ja ens seran més útils de cara a la implementació al programa de simulació:

$$v(n) = v(n-1) + k_u * u(n-1) * \Delta$$

$$y(n) = y(n-1) + v(n-1) * \Delta + k_u * u(n-1) * \frac{\Delta^2}{2}$$

On Δ és el període entre mostres, que recordem que en el cas del nostre sistema té un valor de 20,00 ms.

Altres factors que també s'han considerat per tal de tenir un model més proper a la realitat han estat els següents:

1) Retard d'n mostres: degut a diverses causes, és possible que el sistema presenti un retard que fa que el controlador doni resposta als estímuls amb un cert desfasament que es pot traduir en un nombre de mostres concret. La manera com això es tradueix a detalls d'implementació és canviant els termes $(n-1)$ per termes $(n-1-m)$, on m és el retard en mostres que experimenta el sistema.

2) Constant de temps τ per a l'acceleració: a causa de la inèrcia dels motors físics, un canvi sobtat en la comanda de velocitat de rotació no es tradueix en un canvi immediat d'aquesta magnitud, sinó que aquesta creix o decreix amb una certa constant de temps anomenada tau, τ. Per a implementar aquest efecte en la simulació s'han afegit termes extres a les línies corresponents al càlcul de $v(n)$ i $y(n)$ que tenen en compte aquesta variació progressiva del valor de l'acceleració.

   La manera com es va determinar de manera aproximada el valor d'aquesta constant de temps va ser utilitzant un muntatge similar al descrit anteriorment (en l'apartat de mesura de les freqüències de rotació del motor en funció de l'entrada) per a analitzar en el domini temporal la variació del període de les ones auditives. Els resultats obtinguts amb el programa d'*Octave* desenvolupat específicament per aquest propòsit, que es poden observar a la figura 4.5, mostren que el valor de τ és proper al valor del període entre mostres, de l'ordre de 20,00 ms.

3) Soroll del sensor i filtrat d'aquest: el sensor utilitzat en el dron té un cert soroll que s'ha modelat per a la simulació com a soroll Gaussià de potència variable que es suma a la posició calculada pel bucle de càlcul del control. Tal com s'ha fet en la implementació del control al dron, a la simulació s'ha implementat un filtre de mitjana mòbil de mida parametritzada que es computa també en aquest mateix bucle de control.

Aquests tres factors també s'han tingut en compte a l'hora d'implementar l'algorisme de simulació en *Octave*, i conjuntament amb la resta d'informacions que s'han arribat a conèixer sobre la planta són els que detallen el model de la planta que s'utilitzarà en el següent apartat observar els efectes de determinats canvis en els paràmetres del control a la planta.

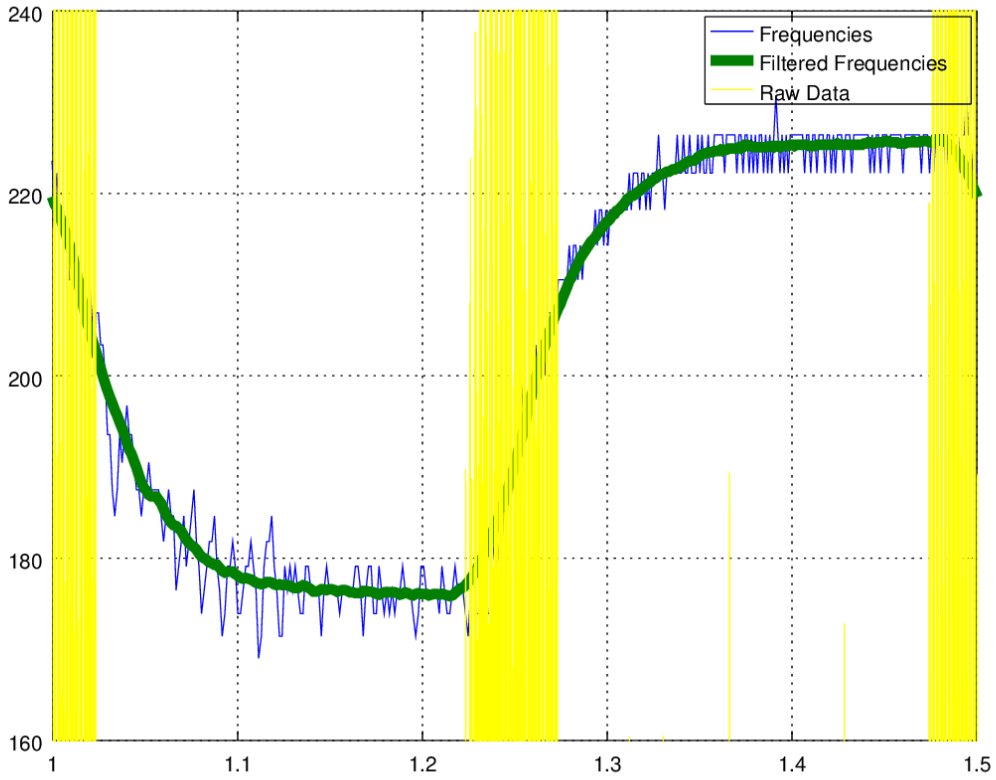

Figura 4.5: Gràfic en el domini temporal que mostra la constant de temps associada als canvis de velocitat del motor. En groc, les dades sense filtrar que marquen els instants corresponents a canvis de velocitat per mitjà de tons emesos per un brunzidor.

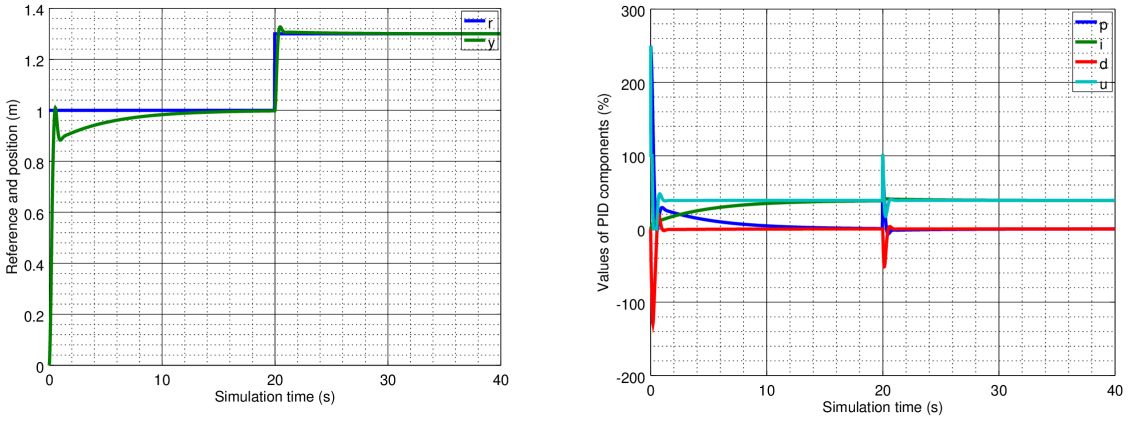

Figura 4.6: Simulació sense factors no ideals amb els paràmetres del control ajustats.

4.2.2 Simulació del control

L'ajustament de les constants dels components `P`, `I` i `D` d'un controlador PID és un procés del qual n'acaba depenent en gran manera la qualitat dels resultats finals. Si bé des d'un punt de vista analític aquests coeficients es podrien calcular en funció de les característiques del sistema i del resultat desitjat, com ja s'ha comentat això sovint no és possible en escenaris reals com el nostre, en què no es té un coneixement del tot afinat del comportament de la planta. En aquests casos, sovint es recorre a mètodes anomenats heurístics, que defineixen una sistemàtica a seguir per anar ajustant els valors d'aquests paràmetres de manera progressiva. En el nostre cas, per aconseguir tenir uns valors per als paràmetres del controlador PID funcionals, es va seguir el procediment descrit a l'article [**wescott**]. Amb això es van arribar a uns valors que, sense introduir ni soroll ni retards, són capaços de controlar el sistema de manera prou satisfactòria, tal com es pot observar a la figura 4.6.

Utilitzant aquests paràmetres com a punt de partida, i repetint sempre la simulació amb la mateixa entrada (la referència consisteix en un valor inicial que experimenta un salt a la meitat de la simulació), anem incorporant elements no ideals dels explicats a l'apartat anterior per estudiar-ne l'impacte.

Simulació amb retard

Per a estudiar l'efecte d'incorporar retards al sistema, augmentem els valors dels paràmetres m i τ, definits com la quantitat de mostres de desfasament i la constant de temps dels canvis d'acceleració, respectivament. A les figures 4.7 i 4.7 podem observar l'efecte d'augmentar el retard del sistema.

Com es pot observar, l'augment dels retards té un impacte directe en l'estabilitat del sistema, ja que la falta de capacitat de reacció del control el fa entrar en un estat marginalment estable que manté el dron en oscil·lació. Portat a l'extrem, s'observa que un retard excessiu acaba duent el sistema a la inestabilitat, com passa a la figura 4.9.

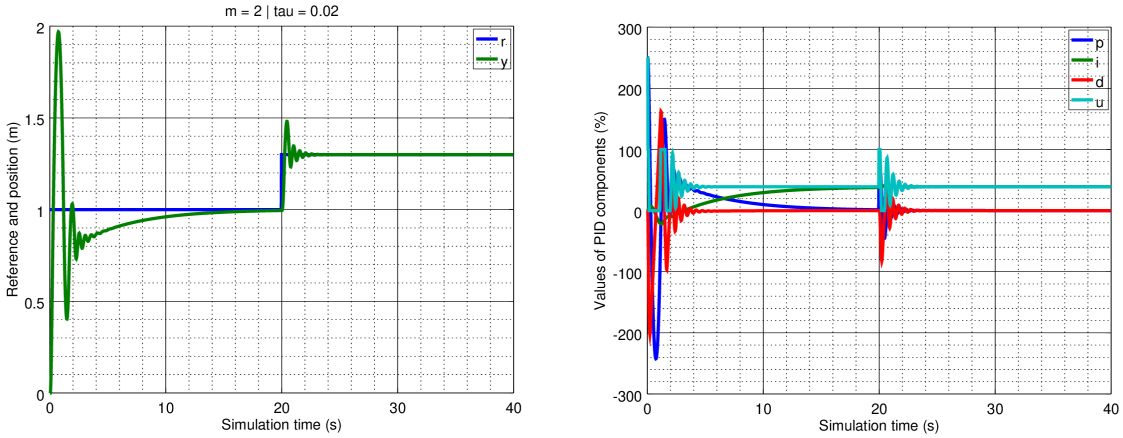

Figura 4.7: Simulació amb retard $\tau = 0.02$ i $m = 2$. El sistema continua sent estable, si bé es degrada la seva operació.

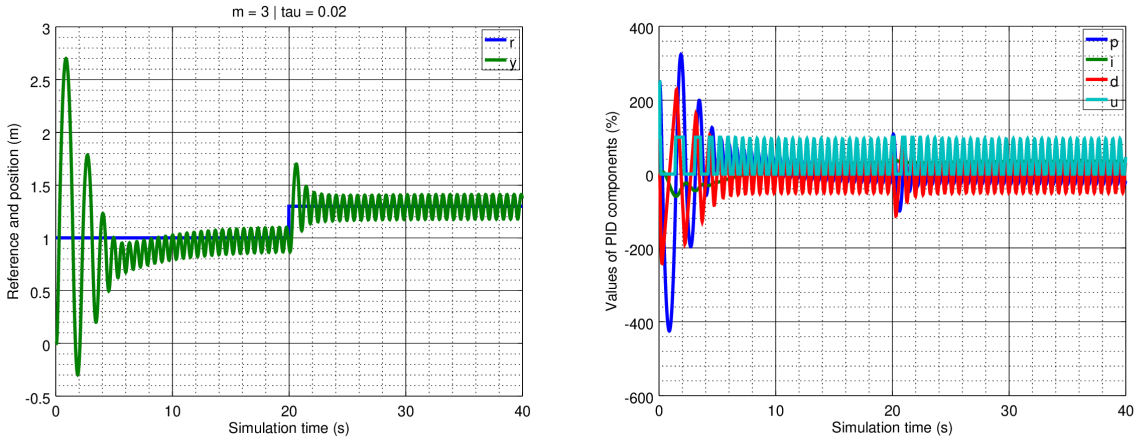

Figura 4.8: Simulació amb retard $\tau = 0.02$ i $m = 3$. S'observa com el sistema és marginalment estable.

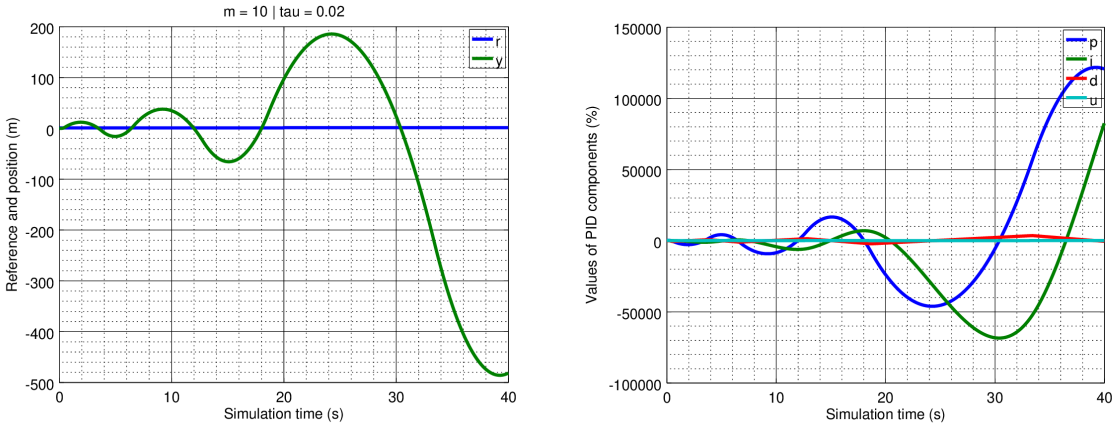

Figura 4.9: Simulació amb retard $\tau = 0.02$ i $m = 10$. S'observa com el sistema es desestabilitza.

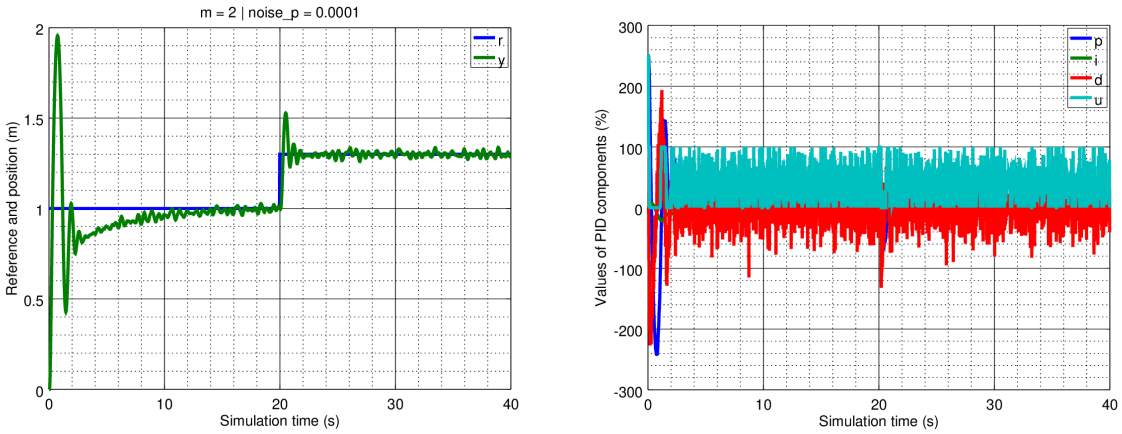

Figura 4.10: Simulació amb soroll de potència $noise_p = 0.0001$.

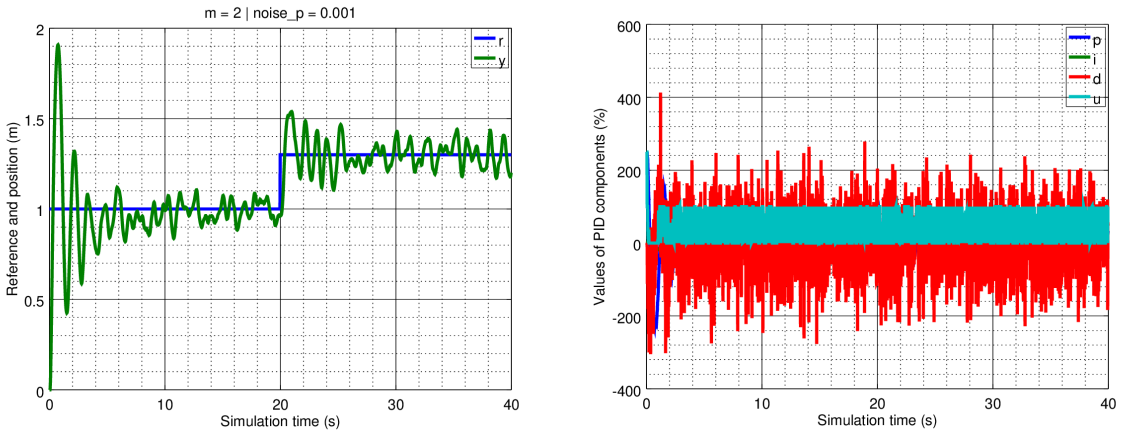

Figura 4.11: Simulació amb soroll de potència $noise_p = 0.001$.

Simulació amb soroll

Per a estudiar l'efecte d'incorporar soroll al sistema, augmentem el valor de la potència de soroll Gaussià que interfereix en el senyal de la posició que actua com a entrada del control (imitant el camí real del soroll, que prové de la realimentació del sistema que permet el sensor, causant del soroll en primer lloc). Cal destacar que partim de l'escenari amb retard mínim que s'ha mostrat en primer lloc a l'apartat anterior, per anar acumulant no linealitats i per tant acostant-nos al model més proper a la realitat. A les figures 4.10, 4.11 i 4.12 es pot observar l'efecte d'anar incrementant la potència del soroll en el senyal.

Igual com passava abans amb el retard, s'observa que l'impacte d'una linealitat com el soroll, en aquest cas, creix a mesura que creix la seva potència, arribant a degradar per complet l'acció del control. Si el soroll arriba a superar un cert llindar, es pot observar com el sistema no pot arribar a l'estabilitat i s'acaba desestabilitzant, com passa a la figura 4.13.

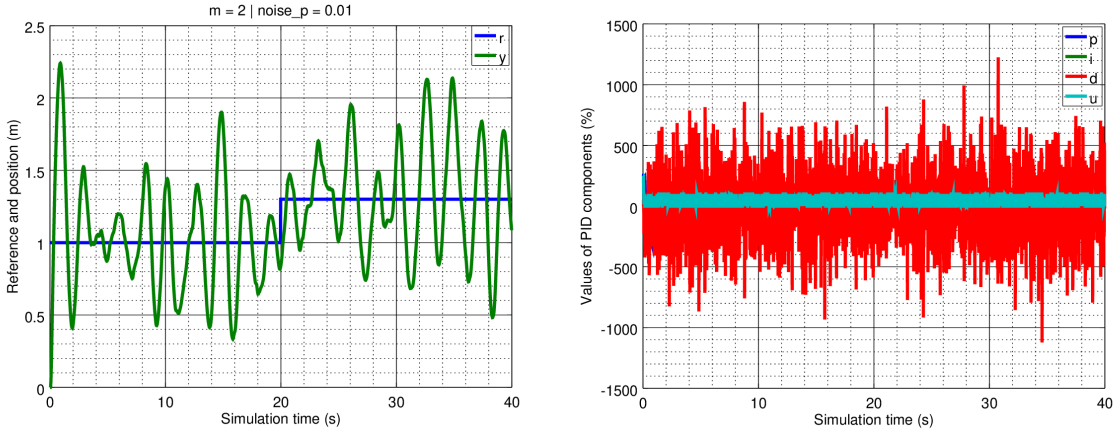

Figura 4.12: Simulació amb soroll de potència $noise_p = 0.01$.

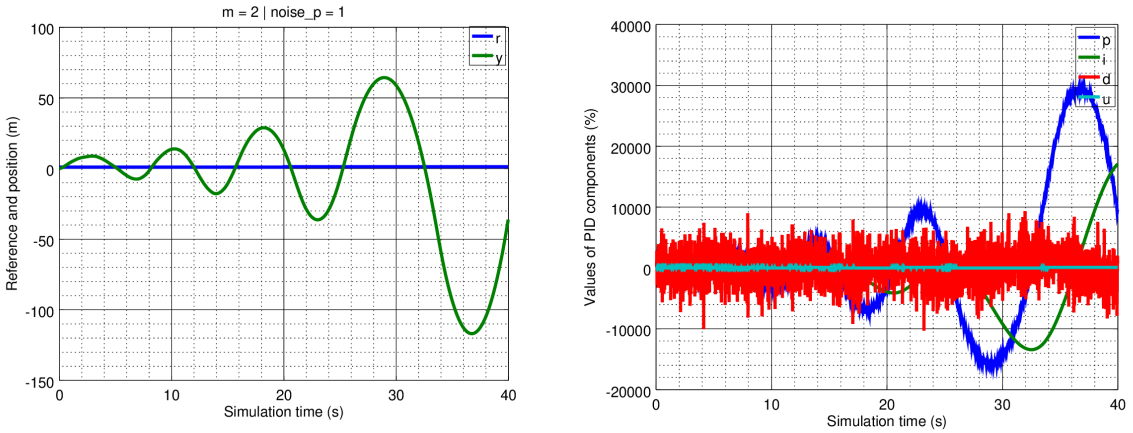

Figura 4.13: Simulació amb soroll de potència $noise_p = 1$. S'observa com el sistema es desestabilitza.

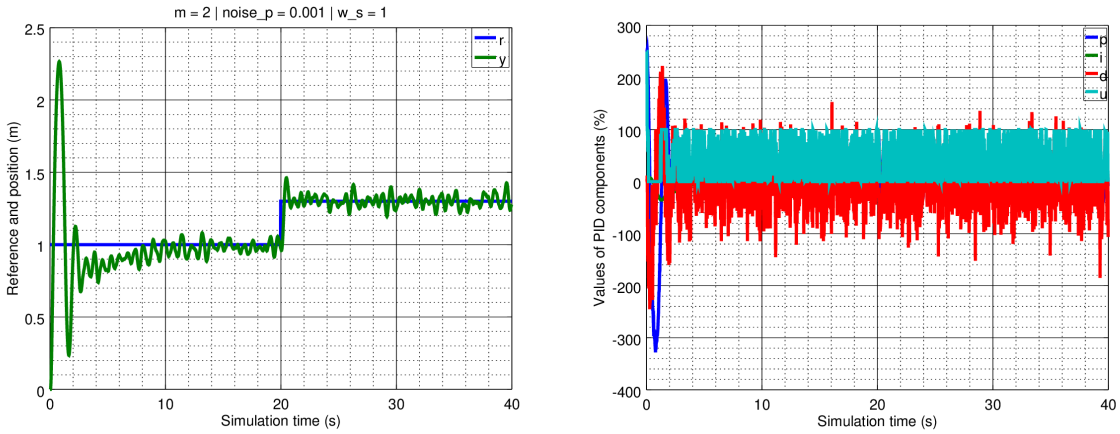

Figura 4.14: Simulació amb mida de finestra $w_s = 1$ respecte de la mostra actual. Es pot observar una reducció de soroll que en abscència del filtre

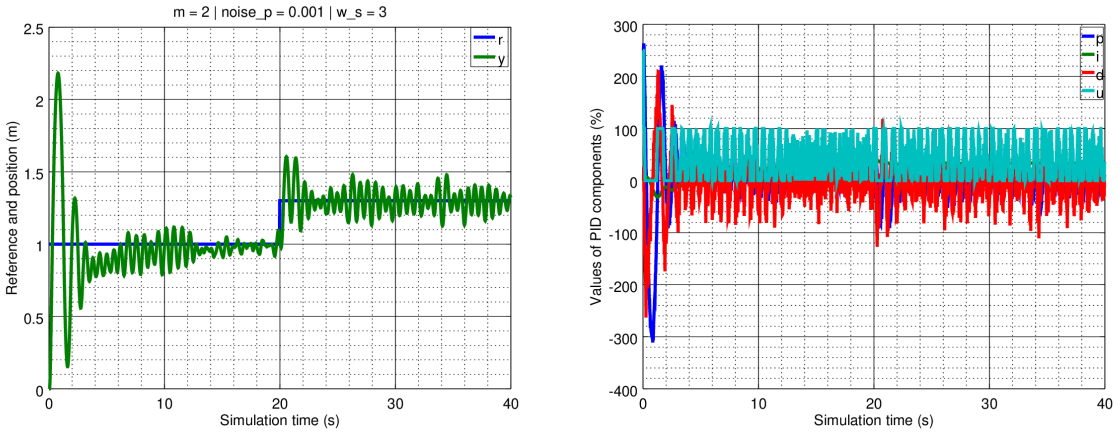

Figura 4.15: Simulació amb mida de finestra $w_s = 3$ respecte de la mostra actual.

Simulació amb filtratge

Per a estudiar l'efecte del filtratge del senyal al sistema s'incrementa progressivament la mida de la finestra de la mitjana mòbil que realitza el filtre i s'utilitza el senyal filtrat com a entrada del component derivatiu del controlador PID. En aquest cas partim de la segona simulació amb soroll, amb una potència de $noise_p = 0.001$, ja que s'ha cregut que és més representatiu del sistema real (sobretot quan es troba a una distància més gran del terra). A les figures 4.14 i 4.15 s'observa aquest impacte.

Es pot observar que l'ús d'una finestra mòbil petita efectivament aporta millores pel que fa a la reducció del soroll, com es pot comprovar comparant la figura amb filtrat 4.14 amb la figura 4.11, que té la mateixa potència de soroll però uns pitjors resultats. Així i tot, també s'observa que tenir una finestra massa gran degrada el resultat a causa de la introducció d'un retard extra.

Un cop més observem que si el filtratge es porta al límit, es corre el risc de desestabilitzar el sistema per culpa del retard extra introduït per la finestra mòbil. Aquest efecte es pot observar

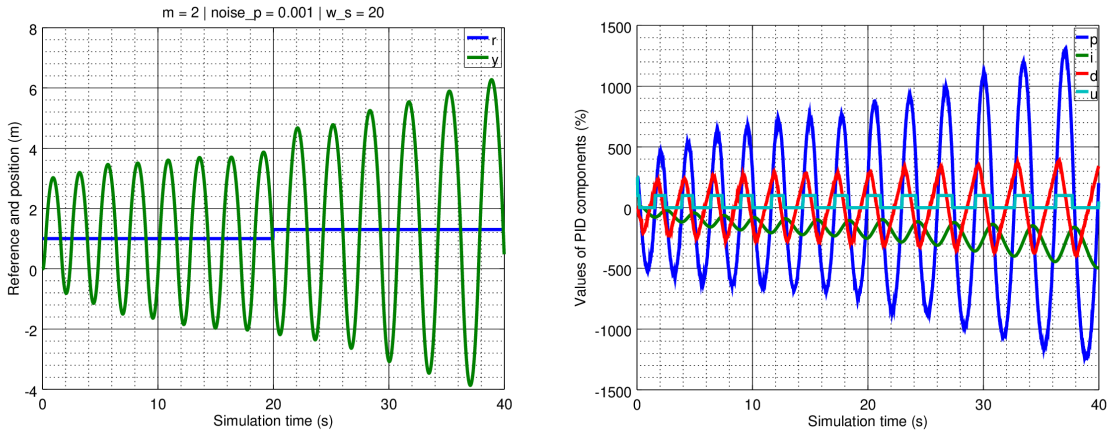

Figura 4.16: Simulació amb mida de finestra $w_s = 20$ respecte de la mostra actual. S'observa com el sistema s'inestabilitza.

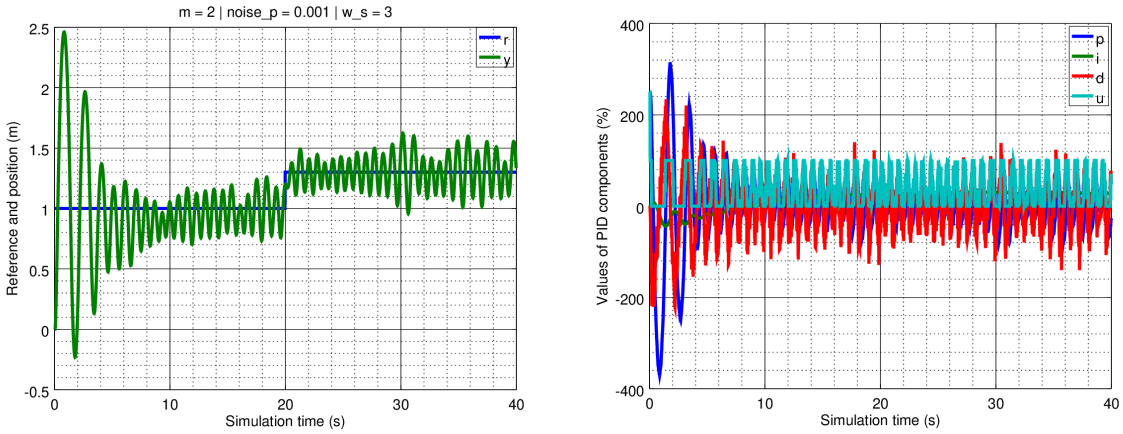

Figura 4.17: Simulació amb mida de finestra $w_s = 3$ respecte de la mostra actual. En aquest cas el senyal filtrat s'aplica a tots els components del control, fet que empitjora els resultats.

a la figura 4.16.

També és interessant remarcar un aprenentatge fet com a part de l'experimentació amb la planta. Si el senyal filtrat s'utilitza com a entrada no només del component derivador, sinó a tot el control, s'observa que de fet els resultats són pitjors, com es pot observar comparant la figura 4.15 amb la figura 4.17.

Simulació amb limitació del senyal de control

Una última linealitat que no s'havia comentat explícitament fins ara és la limitació del senyal de control que imposen per qüestions físiques els motors. Aquest factor ha influït en totes les simulacions, ja que així s'ha considerat. El que no s'ha intentat aplicar fins ara és una limitació encara més estricta, que limiti el valor del senyal de control $u(n)$ a valors propers al punt d'equilibri (moment en què el component proporcional equival a l'acceleració de la gravetat).

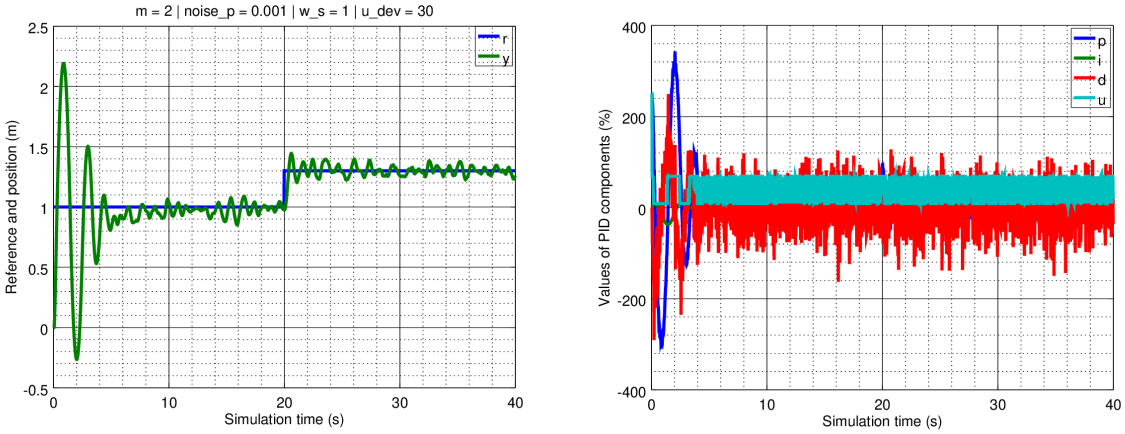

Figura 4.18: Simulació amb limitació del senyal de control a valors dins del rang $u_0 \pm 30$. S'observen millores en comparació a l'abscència de la limitació.

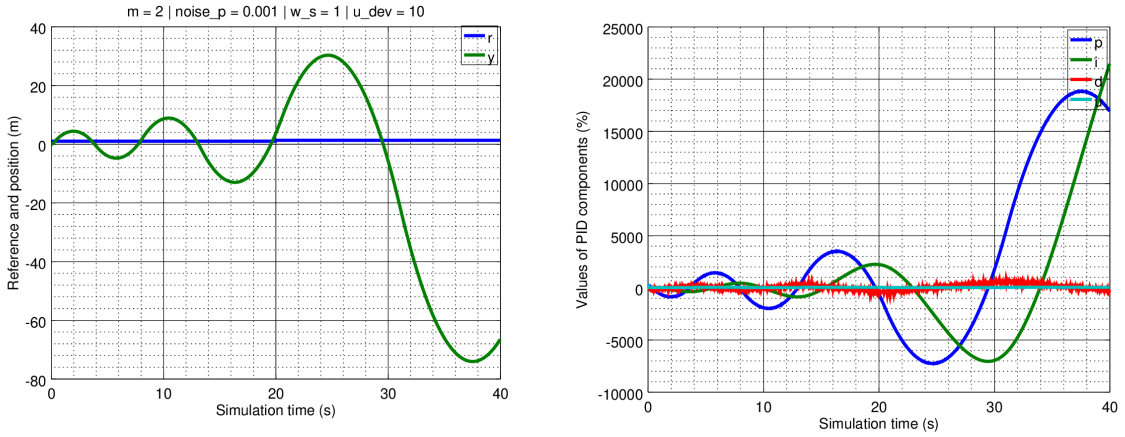

Figura 4.19: Simulació amb limitació del senyal de control a valors dins del rang $u_0 \pm 10$. S'observa com el sistema es desestabilitza.

Al realitzar proves aplicant aquesta limitació partint de la simulació amb un filtratge de mida prudent, s'observa que de fet limitar el senyal de control a valor dins d'un rang de desviació determinat al voltant del punt d'equilibri (que anomenem u_0) millora els resultats, com es pot observar a la figura 4.18.

Un cop més, però, portar aquesta limitació al límit pot desestabilitzar el sistema, ja que reduir massa el rang de valors possibles per al senyal de control pot anul·lar la capacitat d'actuar al controlador sobre la planta, com es pot observar a la figura 4.19.

Cal ser conscients que aquesta estratègia consistent en limitar els valor del senyal d'entrada als actuadors que calcula el control d'alguna manera altera l'estructura del controlador PID en si, ja que modifica el comportament que aquest acaba tenint i el seu efecte sobre la planta. El mateix passa amb el filtratge, ja que fer diferències pel que fa al senyal d'entrada dels diferents components del controlador suposa un canvi respecte de l'estructura tradicional del controlador PID que trenca fins a cert punt amb els conceptes subjacents a aquest.

Tot i això, s'entén que en determinades ocasions la simplicitat o comoditat d'utilitzar un controlador de funcionament preestablert com és un PID pot no donar els resultats esperats, i en aquests casos no es veu com una mala pràctica la de realitzar petits ajustos que puguin millorar el funcionament del sistema, si bé això requereix primer una etapa de familiarització i coneixement de la planta: les tècniques que s'han aplicat per a millorar el funcionament del sistema no s'haurien pogut implantar sense conèixer, per exemple, el punt d'equilibri de la planta, el màxim retard que pot assumir o el nivell de soroll al sensor.

4.2.3 Avaluació dels resultats

Després de les proves realitzades amb el programa de simulació, podem concloure que s'han extret diversos aprenentatges que si bé no s'apliquen de manera quantitativa a la planta real, qualitativament ens poden ajudar a saber els canvis a fer al controlador PID "pelat" per tal d'aconseguir el comportament desitjat pel sistema. Aquests aprenentatges són:

1) Per a determinar els paràmetres base del controlador PID és recomanable utilitzar mètodes heurístics que permeten assolir uns resultats prou bons en unes poques iteracions de canvis al sistema

2) Totes les fonts de retard del sistema s'han de tenir controlades, ja que poden arribar a desestabilitzar-lo

3) El soroll del sensor és un problema perquè degrada el funcionament del sistema quan aquest es troba en estat estable

4) Per a reduir aquest soroll es pot aplicar un filtratge del senyal d'entrada del component derivador, i només d'aquest, vigilant que el retard extra no desestabilitzi el sistema

5) Limitar el rang de valors que pot adoptar el senyal de control del sistema pot millorar els resultats tot suavitzant la capacitat d'actuar sobre la planta que té el controlador

Així doncs, el següent pas lògic és configurar el dron per tal de tenir a la maqueta un control que faci que el comportament del sistema sigui l'esperat, incorporant els canvis necessaris després d'haver estudiat els resultats de les diferents simulacions.

A continuació es mostraran fragments de mostres obtingudes amb el programa de l'estació terrestre que mostren el comportament del dron donats diferents paràmetres, per mirar de comparar els resultats obtinguts a la planta real amb els obtinguts per mitjà de les simulacions.

En primer lloc, el comportament de la planta sense aplicar cap limitació ni filtratge es pot observar a la figura 4.20. Com es pot veure es tracta d'un sistema clarament inestable que no compleix amb els requeriments marcats per al control.

Així doncs, primer apliquem la limitació del valor del senyal de control, i els resultats es poden observar a la figura 4.21. Es pot observar que no es beneficiós pel control ni que la limitació sigui massa laxa (en aquest cas seguim tenint un *overshoot* o sobrepàs relativament gran) ni massa estricte (en aquest cas el temps que es tarda a assolir la consigna creix considerablement).

Després d'experimentar s'arriba a considerar que un valor òptim per al rang de la limitació del senyal de control és de $u_{dev} = 10$. Amb aquest paràmetre fixat, passem a considerar la mida òptima per a la finestra del filtre de mitjana mòbil. Després de també diverses iteracions s'acaba establint que la mida òptima per aquesta finestra, que permet reduir el soroll però sense acostar-se excessivament al punt d'inestabilitat, és de $w_s = 6$. Els resultats obtinguts

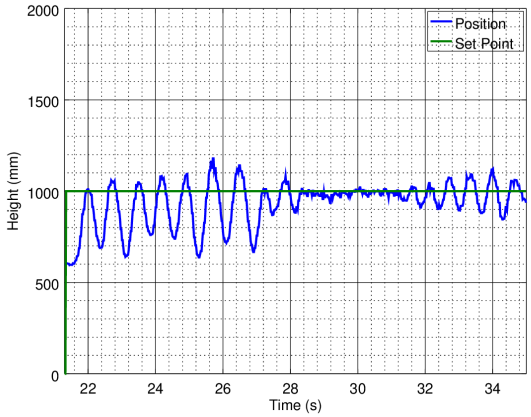

Figura 4.20: Simulació del comportament del dron sense limitació ni filtratge. Es tracta d'un sistema clarament inestable.

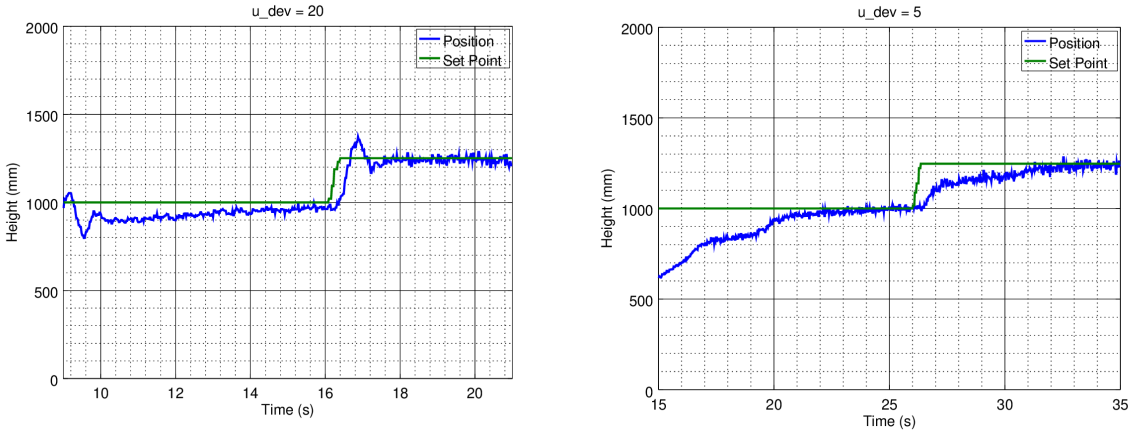

Figura 4.21: Simulació del comportament del dron amb limitació massa laxa i massa estricte.

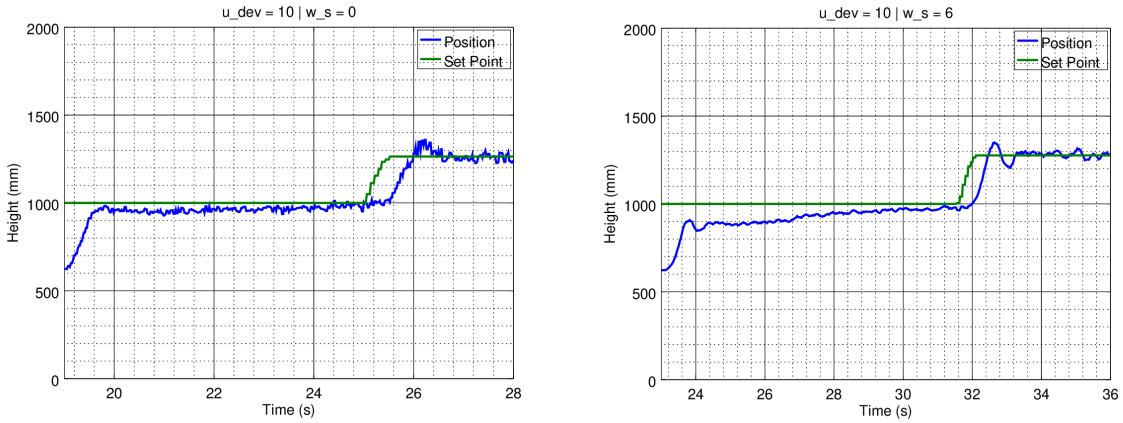

Figura 4.22: Simulació del comportament del dron ja limitat sense filtre i amb filtratge.

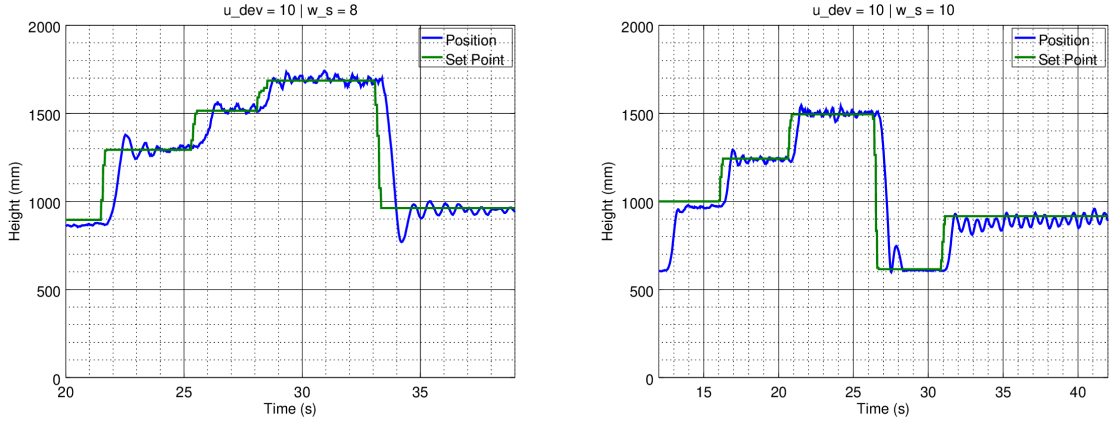

Figura 4.23: Simulació del comportament del dron ja limitat amb filtratge amb finestra creixent. Es pot observar com l'estabilitat del dron es veu compromesa per culpa del retard introduït.

amb aquest filtre comparats amb els obtinguts quan no es realitza cap filtratge es poden observar a la figura 4.22.

Es pot apreciar que hi ha una millora substancial, sobretot pel que fa a la reducció del soroll. Tot i això, un cop més, cal tenir present que aquesta finestra no pot ser més gran per risc a perdre estabilitat, com es pot observar que passa a la figura 4.23. En aquest cas s'observa que introduir més retard ampliant la mida de la finestra s'arriba a tenir més oscil·lacions fins al punt de poder arribar a una situació d'estabilitat marginal (que possiblement només ho és gràcies a les limitacions del senyal de control que eviten que es desestabilitzi per complet).

Finalment, podem dir que gràcies als conceptes deduïts de les simulacions realitzades i gràcies a l'experimentació sobre la maqueta s'ha arribat a parametritzar un sistema de control que permet controlar el dron d'acord amb el què es comentava al principi del capítol: som capaços de donar una consigna d'alçada al dron i aquest la segueix amb una precisió i rapidesa considerables donades les limitacions de la maqueta. A la figura 4.24 es pot observar la resposta del dron al llarg d'una mostra amb diversos canvis de consigna que demostra que s'han assolit

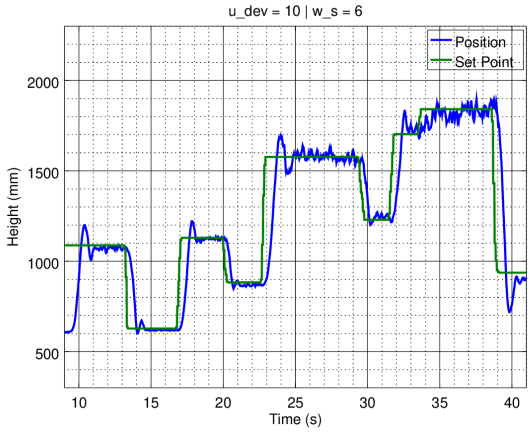

Figura 4.24: Mostra d'una sessió de vol del dron amb el control ajustat. Es pot observar com el comportament del sistema compleix amb els objectius marcats.

els objectius del control.

Val la pena comentar els diferents factors que han propiciat que, si bé qualitativament els resultats pràctics han coincidit amb els obtinguts per mitjà de les simulacions, els resultats numèrics tinguin diferències importants. Entre altres factors, molts dels quals possiblement ni s'han arribat a comprendre ni tenir en compte, alguns dels que han causat diferències entre la teoria i la pràctica han estat les no linealitats que s'han comentat al llarg del capítol. Per mencionar-ne algunes:

1) Mal modelat del soroll: com s'ha comentat a l'apartat corresponent al sensor que incorpora el dron, el soroll d'aquest s'accentua a mesura que creix la distància entre el sensor i els objectes detectats. Aquest fet no s'ha arribat a tenir en compte a les simulacions, ja que s'ha considerat un soroll Gaussià la potència del qual no varia en funció de l'alçada del dron.

2) Alguns aspectes que no s'han tingut en compte i s'han menyspreat, com per exemple el fregament (que pot variar en funció de les vibracions del dron) o el pes del cable (que varia en funció de l'alçada, ja que passa a penjar més o menys).

3) Les mesures de velocitats de rotació, forces d'empenyiment i fregament o els retards i constants de temps s'han aproximat, i per tant els valors obtinguts disten de tenir validesa numèrica, fet que compromet els resultats quantitatius que es pugui obtenir per mitjà de les simulacions.

Aquests aspectes que han dificultat l'obtenció de resultats quantitatius vàlids fan palès el grau de complexitat que amaga el modelatge d'una planta real, subjecta a un gran nombre de variables que fan que sovint s'acabi descartant aquesta opció (la d'arribar a paràmetres de control per mitjans analítics) a favor de procediments més heurístics i empírics com els que s'han aplicat en el nostre projecte.

5 Conclusions i treball futur

A tall de conclusions per aquest projecte, es farà una revisió dels objectius plantejats a l'inici d'aquest per valorar en quin grau s'han assolit i per posar èmfasi en els resultats obtinguts al final del treball.

Així doncs, recuperem els objectius plantejats al començament de la memòria:

1) Construir una maqueta de control basada en un dron

2) Desenvolupar el conjunt d'eines *software* que permetin monitorar i controlar el comportament del sistema des d'una estació terrestre

3) Implementar i estudiar, mitjançant les eines desenvolupades, un sistema de control funcional per a la maqueta construïda

Per la manera com es van plantejar aquests objectius, d'alguna manera l'assoliment de cada un d'ells requereix que s'hagin assolit també els objectius anteriors, ja que cada un d'ells suma als resultats obtinguts amb el compliment d'aquests. I efectivament, podem dir que aquest ha estat el nostre cas, ja que s'ha aconseguit construir la maqueta de control basada en un dron, s'han arribat a desenvolupar les eines necessàries per a governar el sistema des d'una estació terrestre i s'han utilitzat aquests dos elements de manera conjunta per implementar un esquema de control que ha permès governar la planta al mateix temps que s'ha pogut estudiar i aprofundir en el camp de la teoria de control de manera pràctica.

Al mateix apartat on es definien els objectius també es parlava d'una visió concreta, nascuda de la motivació per aquest projecte en primer lloc, que buscava en definitiva perseguir la creació d'una plataforma docent que permeti fer exactament allò que s'ha fet com a part del treball: aprendre sobre els mecanismes i conceptes relacionats amb el control però d'una manera més vivencial i fins i tot engrescadora. Així doncs, es considera que mentre es pugui continuar utilitzant aquesta plataforma com a eina d'interès acadèmic, s'han assolit tots els objectius planejats i s'ha complert la visió inicial per al projecte.

Així i tot, és innegable que el treball no ha arribat a tancar totes les vies d'ampliació i millora del projecte o de la plataforma en si mateixa, per la qual cosa es poden mencionar encara algunes línies de treball futur que es creu que podrien ser interessants. Aquestes línies de treball tenen dos vessants diferents, una de les quals té a veure amb els aspectes més tècnics del treball mentre que l'altra fa referència a les oportunitats que ofereix l'eina desenvolupada. Així doncs, a continuació tenim una llista no exhaustiva de possibles fils que es podrien seguir per a ampliar el treball fet:

1) *Vessant tècnica:* aquest vessant del treball futur obre la porta a millores en la realització de les eines desenvolupades, així com de l'anàlisi que s'ha fet d'aquestes.
    a) *Maqueta:* La maqueta física queda oberta a possibles millores que busquin fer-la més robusta, revisant per exemple el disseny del suport o del dron en si (el sistema de fregament entre el dron i la guia es podria millorar en gran manera) fins i tot

arribant a permetre, en un futur, que aquest pugui tenir més d'un grau de llibertat (afegint eixos de gir, per exemple).

b) *Programari:* els programes desenvolupats tant pel dron com per l'estació terrestre es podrien ampliar per fer-los més flexibles, robustos i eficients, fet que podria permetre (acompanyat d'una revisió de l'electrònica que inclou el dron) treballar a velocitats més elevades i així poder implementar un control més funcional. També es podria treballar l'accessibilitat d'aquests programes, permetent configurar més aspectes del funcionament del dron o fins i tot canviar el control a través d'una interfície gràfica a partir de blocs (en un estil semblant al que permeten eines més potents com ara *Matlab*).

c) *Simulació i control:* Els scripts encarregats de realitzar les simulacions que s'han fet com a part d'aquest projecte podrien incorporar mecanismes més sofisticats per a definir les condicions de simulació (fins ara s'ha fet de manera manual) i així, acompanyat d'un procés d'estudi a fons del comportament de la planta que permeti obtenir un model més fidel a la realitat, poder arribar a simular escenaris que corresponguin amb resultats obtinguts de manera analítica, ampliant les possibilitats d'aprenentatge sobre el camp del control.

2) *Vessant docent:* aquest vessant del treball futur obre la porta a la recerca de les possibles maneres d'aprofitar aquesta eina per extreure'n el màxim profit des d'un punt de vista acadèmic.

a) Integració de la maqueta com a part del material docent per a assignatures relacionades amb els àmbits TIC: la plataforma desenvolupada es podria considerar com una part dels materials docents d'algunes assignatures del grau orientades a posar en pràctica els coneixements sobre programació, protocols de comunicació o teoria de control, òbviament.

b) També es podria realitzar un estudi per determinar aquells conceptes sobre teoria de control que millor es poden il·lustrar utilitzant la plataforma creada, per així tenir un suport extra a l'hora d'ensenyar-los que agilitzi l'aprenentatge i permeti als estudiants anar més enllà del pla d'estudis a través de l'experimentació amb la maqueta de manera autònoma.

c) Aquesta maqueta de control basada en un dron obre la possibilitat a explorar aquesta tecnologia, la dels vehicles aeris no tripulats, pel que fa a aspectes que fins ara potser no s'havien considerat per falta de proximitat amb el sector. La UPC forma part de competicions en l'àmbit internacional en què equips d'estudiants d'enginyeria desenvolupen cotxes per mirar de guanyar curses. Per què no iniciar un projecte semblant però que explori el sector dels drons, una tecnologia en creixement i altament popular?

I amb aquestes propostes de possibles línies de treball futur que altres estudiants o enginyers podrien decidir explorar més endavant, i amb el desig que la plataforma creada com a resultat d'aquest projecte sigui útil per als que ens seguiran, es dóna per conclòs aquest projecte de final de grau d'Enginyeria en Sistemes TIC.